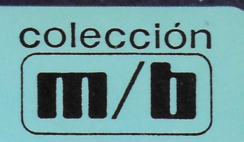
SERIE
AUTOAPRENDIZAJE
ACELERADO



introducción a la

inteligencia artificial con el spectrum

keith y steven brain

el microordenador convertido en un equipo "inteligente"

REDE

Introducción a la Inteligencia Artificial con el Spectrum

KEITH Y STEVEN BRAIN

Introducción a la Inteligencia Artificial con el Spectrum

EDICIONES TECNICAS REDE, S. A.
Ecuador, 91 - Tlfno. 250.30.97
08029 BARCELONA

Título original: ARTIFICIAL INTELLIGENCE
ON THE SPECTRUM COMPUTER

Editado en inglés por:
Sunshine Books (imp. Scot Press Ltd.)
12-13 Little Newport Street
Londres WC2R 3LD

Traducción/Adaptación: Alfonso Martínez María

Todos los programas han sido comprobados en el Departamento de Microinformática de EDICIONES TECNICAS REDE, S. A.

ISBN: 84-247-0222-0

Impreso en España — Printed in Spain

Dep. Legal: B. 42281-85 — REDEPRINT — Barcelona

SUMARIO

Introducción

La Inteligencia Artificial es, sin duda, un campo de creciente importancia en el desarrollo de los ordenadores que tendrá efectos profundos en nuestras vidas en las próximas décadas. El objetivo principal de este libro es introducir al lector en algunos de los conceptos relacionados con la Inteligencia Artificial y mostrarle cómo elaborar rutinas «inteligentes» en el BASIC de Sinclair que puedan después incorporarse a sus propios programas. Se supone que sólo se posee un conocimiento superficial del BASIC y el libro parte de principios fundamentales ya que pensamos que ello es esencial para conocer los problemas de la creación de inteligencia y la forma de resolverlos.

La organización básica del libro se apoya en la elaboración de las ideas y de las rutinas consiguientes, paso a paso, explorando y comparando las posibilidades alternativas siempre que es posible. En lugar de limitarnos a proporcionar una serie de programas completos, preferimos estimular al lector a que experimente con enfoques diferentes para que pueda ver los resultados por sí mismo. Se incluyen diagramas de flujo detallados de la mayoría de las rutinas, destacando los aspectos de Inteligencia Artificial y hemos evitado, por consiguiente, recargar las presentaciones en pantalla ya que, de esta forma, se oscurece el significado del programa. En algunas ocasiones se advertirán líneas redundantes que se han incluido deliberadamente para la mejor claridad del programa. Se ha evitado todo lo posible la reconstrucción de las líneas habiéndose preferido hacer modificaciones en ellas. Todos los listados se presentan en la misma forma en que han de verse en la pantalla. En la mayoría de los casos se han utilizado profusamente espacios y paréntesis para facilitar la lectura de los listados aunque ha de tenerse cuidado si se piensa en

suprimirlos ya que algunos son esenciales. Se han utilizado letras mayúsculas pues pensamos que así se leen mejor los listados pero, si se desea, pueden utilizarse igualmente las minúsculas. Recuérdese, sin embargo, que las cadenas en mayúsculas no son comparables con las que están en minúsculas por lo que han de manejarse en forma coherente. Todas las rutinas han sido comprobadas rigurosamente, así como el listado en general, por lo que confiamos en que no se encontrarán fallos. Es un hecho triste y real en la vida que la mayoría de los fallos se derivan de errores de escritura. Los puntos y coma, así como las comas, pueden parecer elementos del léxico insignificantes pero su ausencia produce efectos profundos.

La Inteligencia Artificial está adquiriendo más importancia cada día y esperamos que este libro proporcione al lector una visión útil del tema y si realmente lo trabaja en profundidad, ¿quién sabe?, ¡a lo mejor es capaz de persuadir a su máquina a que lea por sí misma nuestro próximo libro!

Nuestro sincero agradecimiento a Valerie James que escribió y probó la mayor parte de los programas que fueron trasladados de los originales para el ordenador Commodore 64/Dragón al BASIC de Sinclair para esta versión. También tenemos que manifestar nuestro reconocimiento una vez más a Liz, que ya sabe sobre ordenadores lo suficiente como para dominar los controles de un horno de microondas para que nuestro café esté a la temperatura correcta mientras ella espera pacientemente a que alguien invente un androide que pueda tirar la bolsa de la basura.

Keith y Steve Brain
Groeswen

1

La Inteligencia Artificial y sus interpretaciones

Durante generaciones los escritores de ciencia ficción han imaginado el desarrollo de máquinas inteligentes que pudieran realizar muchas de las actividades humanas e incluso superarlas en algunos campos. La Inteligencia Artificial se ha visto alentada por estas imágenes. La visión más común de un robot es la de una máquina inteligente, generalmente con forma humana, que es capaz de ejecutar por sí misma las instrucciones que se le dan en términos vagos.

En la opinión pública se ha desarrollado una posición de desconfianza hacia las nuevas tecnologías. Por tanto, no es de extrañar que, en las primeras narraciones sobre estos temas, los robots tuvieran muy mala prensa y desempeñaran siempre el papel de los «malos» tradicionales, con el agravante de ser casi invencibles y manifestar una total falta de escrúpulos. El clarividente Isaac Asimov urdió una larga serie de historias alrededor de su concepto de los «robots positrónicos» y fue probablemente el primer autor que se enfrentó con la realidad de esta situación. Estableció sus famosas «Tres Leyes de la Robótica» que definen las reglas fundamentales según las cuales ha de funcionar cualquier máquina capaz de realizar una acción independiente pero es interesante observar que no pudo predecir en qué momento la especie humana aceptaría la presencia de tales robots en la Tierra.

«La Guerra de las Galaxias» introdujo los robots especializados R2D2 y C3PO aunque creemos que muchas de sus características eran un poco extrañas. Quizás existía una Unión Interplanetaria de Robots

con una normativa de los campos de actividad de los mismos que impedía la comunicación directa entre humanos y R2D2. En «The Stepford Wives» («Las Esposas de Stepford»), los maridos se reúnen y tienen la «buena idea» de convertir a sus esposas en androides que, en forma automática, hagan exactamente lo que se esperaba de ellas. Sin embargo, en la segunda parte de la obra, se revelan los peligros que entraña la exigencia de incrementar constantemente los estímulos externos. Quizás una esperanza para la Humanidad es que cualquier alienígeno que nos amenace no haya visto «Battlestar Galactica» y, por tanto, construya los robots de tipo Cylon que, como los viejos Invasores del Espacio, siempre acaban siendo derrotados, ya que su conducta es totalmente previsible.

Por supuesto que los ordenadores inteligentes también aparecen como cajas sin brazos ni piernas, aunque las luces intermitentes parecen obligatorias. Las entradas y salidas deben ser obviamente orales aunque ya ha pasado de moda la vieja voz metálica sustituida por una personalidad algo más definida. Si todas las cajas se parecen, debe ser una buena idea pero, por favor, no acepten todo el sonido del «Sargento Mayor Cero» de la película «Terrahawks». «Kitt» de Michael Knight («El coche fantástico» en TVE), suena como una especie de máquina razonable con la que conversar, y resulta verdaderamente preferible a la untuosa voz de «Slave» y del aborrecible «Orac» de «Blake's Seven». «Orac» parecía acumular un tremendo desdén en aquella pequeña caja pero otros escritores han apreciado las dificultades que pueden surgir si se hace que la personalidad de una de esas máquinas sea muy parecida a la del hombre.

En «Odisea en el Espacio: 2001», de Arthur C. Clark, el ordenador de máxima inteligencia «Hal» tiene, al final, una «crisis nerviosa» cuando se enfrenta a tantas responsabilidades. Sin embargo, en «Dark Star» la bomba inteligente se sentía feliz discutiendo sobre Existencialismo con el capitán Doolittle, aunque no aceptaba desviarse del tiempo programado de detonación cuando aún se hallaba situada en su alojamiento. En «The Restaurant At The End of The Universe» («El Restaurante del Fin del Universo») se reduce significativamente el valor del «Sirius Cybernetics Corporation Happy Vertical People Transporter» (Transportador Vertical de Gente Feliz de la «Corporación Cibernética Sirio»), cuando rehusa subir ya que podía ver el futuro y si se elevaba probablemente sería atrapado. Y el

Sintetizador de Bebidas «Nutri-Matric» había sido obviamente diseñado por la «British Rail Catering» ya que siempre producía una bebida que era «casi, aunque no enteramente, distinta al te».

También han aparecido recientemente temas más preocupantes. Lo más inquietante de la obra «Wargames» (Juegos de Guerra) no es que alguien pueda introducirse en «Joshua» (el Ordenador del Departamento de Defensa de Estados Unidos) sino que, una vez la máquina ha empezado a jugar a la guerra termonuclear, no se puede parar hasta que haya un vencedor. En «The Forbin Project» («El Proyecto Forbin») los ordenadores de Rusia y de los Estados Unidos se unen y deciden que los humanos son insignificantes. Por supuesto, si usted es Marvin, el Androide Paranoico, y tiene el cerebro del tamaño de un planeta y posee una auténtica personalidad humana, podría lograr la victoria sin armas, confundiendo sencillamente al enemigo para que provoque su propia destrucción mientras trata sus problemas personales.

La debatida definición de la Inteligencia Artificial

La definición y el reconocimiento de la inteligencia de las máquinas han sido debatidos con cierto encrespamiento entre los expertos en el tema. La definición más generalmente aceptada es la que fue propuesta por Alan Turing, a finales de los años 40, cuando los ordenadores eran del tamaño de una casa y más raros de lo que es hoy una regla de cálculo. En lugar de tratar de establecer una serie de criterios a los que someter las definiciones, tomó una visión más amplia del problema. Razonó de la siguiente manera: la mayoría de las personas aceptan el hecho de que sus semejantes, en general, son inteligentes. Por tanto, en el momento en que una persona es incapaz de determinar si se está relacionando con otra persona o con un ordenador, necesariamente ha de aceptar que es inteligente la máquina con la que se relaciona. Esto constituye la base de su famosa «Prueba de Turing», por la que un operador ha de mantener una conversación con otra entidad a través de un teclado y tratar de lograr que la otra parte revele si se trata realmente de una máquina o bien de otro ser humano.

Circulan muchas historias de ficción sobre esta prueba pero la que preferimos es una en la que un aspirante a un cierto trabajo se

sienta frente a un teclado y se le deja solo. Al comprender la importancia que la prueba tiene para su carrera, lucha valientemente para encontrar el secreto, sin conseguir éxito, al menos aparentemente. Sin embargo, después de algún tiempo, el entrevistador le llama y, estrechando su mano, le felicita con estas palabras: «Buen trabajo, amigo, la máquina no ha podido decir si ha tratado con un hombre por lo que usted es lo que necesitamos como Inspector Fiscal de Su Majestad».

La inteligencia mínima de los robots

Todos hemos visto en los anuncios de la televisión que las técnicas de diseño asistidas por ordenador se hallan ahora muy difundidas y que los robots industriales son los únicos protagonistas de las líneas de producción de automóviles (lo que da lugar a los adhesivos que se ponen en las ventanillas de los coches con la expresión: «Diseñado por ordenador, construido por un robot y conducido por un idiota»). En realidad, la mayoría de estos robots industriales son de inteligencia mínima ya que se limitan a seguir unas operaciones predefinidas sin hacer nada que pueda considerarse una toma real de decisiones. Incluso el impresionante robot que pinta por difusión y que se limita a seguir unas pautas aprendidas de los movimientos manuales que un operador humano obliga a realizar a sus brazos mecánicos, no puede aprender a ejecutar otra tarea en un nuevo objeto, sin ulterior intervención humana.

Por otra parte, la siguiente generación de robots tendrá sensores más complicados y mejor programación lo que les permitirá determinar la forma, el color y la textura de los objetos, tomar decisiones más racionales. El que haya visto reportajes de los legendarios concursos de «Micromouse» («Micro-ratón»), donde un repulsivo bicho se escapa para meterse en el centro de un laberinto, no se verá sorprendido por nuestra fe en el futuro del robot inteligente aunque no parece muy importante dotarle de dos brazos y dos piernas.

La Inteligencia Artificial en los sistemas expertos

Otra interesante área en la que se explota la Inteligencia Artificial es en el campo de los sistemas expertos, en muchos de los cuales

se pueden obtener logros tan buenos o mejores que los de los humanos experimentados, especialmente si se piensa en la predicción del tiempo. Estos sistemas pueden ser expertos en cualquier número de cosas pero, en especial, en el campo del diagnóstico y tratamiento médico, donde están adquiriendo creciente importancia. La profesión médica no debe preocuparse demasiado ya que siempre quedará espacio para ellos, puesto que los «ordenadores no podrán asociarse».

Una barrera importante que se opone a una mayor utilización de los ordenadores es la ignorancia y tozudez de los usuarios que sólo leen las instrucciones como último recurso y que esperan que la máquina sea capaz de comprender sus pequeñas peculiaridades. El tratamiento del «lenguaje natural» es, por consiguiente, una de las mayores áreas de crecimiento y la «quinta generación» de ordenadores será mucho mejor recibida por los usuarios.

La mayor parte del trabajo serio sobre Inteligencia Artificial utiliza lenguajes más adecuados que el BASIC (aunque más complejos), tales como el LISP y el PROLOG, que son completamente ininteligibles para el usuario medio y que probablemente no estén disponibles para su microordenador doméstico. Las rutinas en BASIC que siguen no le darán, por consiguiente, la clave para dominar el Mundo aunque le permitirán adquirir una razonable apreciación de las posibilidades y problemas que presenta la Inteligencia Artificial.

2

Ordenes eficaces para el ordenador

Puesto que su ordenador carece completamente de inteligencia sólo se puede conversar con él a base de términos muy sencillos. El primer paso, utilizado en muchos juegos de aventuras no complicados, es disponer de una serie de órdenes predeterminadas para las que hay respuestas fijas. Empecemos por echar una mirada a ciertas direcciones de la brújula en las que tenemos que movernos. A primera vista, la forma más sencilla de programar esto parece ser la de pedir un dato de entrada, mediante la instrucción INPUT al usuario y escribir una línea condicional (IF-THEN) para cada posibilidad (véase el Diagrama de flujo 2.1).

```
  10 REM ELABORACION DEL SIGNO D
E ABRIR INTERROGANTE
  20 REM LA LETRA "I" QUE PRECED
E A LA FRASE INTERROGATIVA SE HA
 DE PULSAR EN LA MODALIDAD DE GR
AFICOS (CAPS SHIFT Y 9). NO OLVI
DE PASAR A CONTINUACION AL MODO
NORMAL (PULSANDO 9)
  30 RESTORE 30: FOR N=1 TO 7: R
EAD A: POKE USR "I"+N,A: NEXT N:
 DATA 2,0,2,14,16,17,14
 100 PRINT '"IQUE DIRECCION?"
 120 INPUT I$
```

```
200 IF I$="NORTE" THEN  PRINT "
NORTE"
 210 IF I$="SUR" THEN  PRINT "SU
R"
 220 IF I$="OESTE" THEN  PRINT "
OESTE"
 230 IF I$="ESTE" THEN  PRINT "E
STE"
 232 REM LA INSTRUCCION QUE SIGU
E ES PARA SALIR DEL CICLO
 235 IF I$="F" THEN  STOP
 250 GO TO 100
```

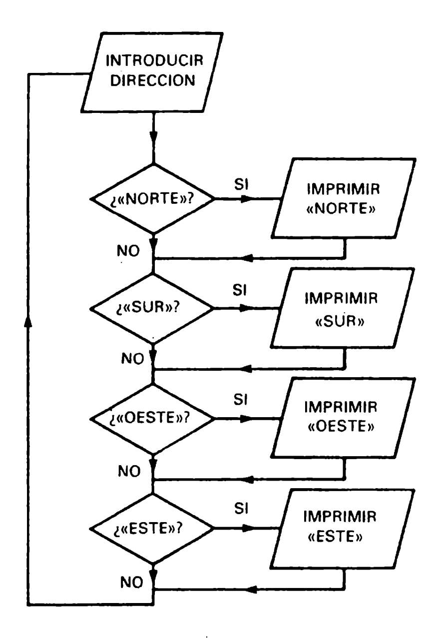


Diagrama de flujo 2.1. Las direcciones de la brújula

Si se hace una entrada distinta a los cuatro puntos cardinales no se imprimirá nada excepto otra petición de datos. Sería mejor que el ordenador indicase con más claridad que la orden no es válida. Esto podría resolverse incluyendo un medio de comprobar que no se ha encontrado el comando dictado pero pudiera ser excesivamente largo

e incluso imposible cuando haya una lista con demasiadas palabras válidas.

```
240 IF I$<>"NORTE"AND I$<>"SUR"
AND I$<>"OESTE"AND I$<>"ESTE"THE
N PRINT "PETICION NO VALIDA"
```

Añadiendo, por otra parte, GO TO 100 al final de cada línea con la instrucción condicional IF-THEN, se forzaría un salto atrás a la primera que solicita datos cuando el ordenador detectase una orden realizable. Si ninguna de las condiciones se cumple, se llega a la línea 240 que hace la advertencia de la invalidez de la orden. Los retornos directos cuando se encuentra la palabra es una buena idea de cualquier modo ya que evita que se hagan búsquedas innecesarias cuando ya se ha encontrado lo que se buscaba (véase el Diagrama de flujo 2.2).

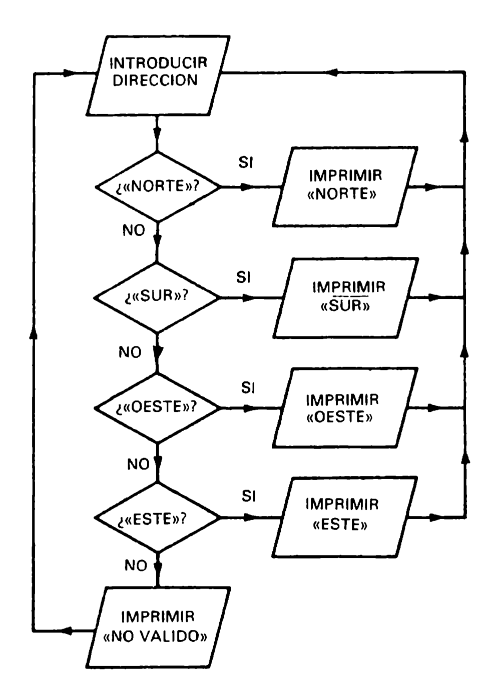


Diagrama de flujo 2.2. Eliminación de pruebas innecesarias

```
 200 IF I$="NORTE"THEN PRINT "NO
RTE":GO TO 100
 210 IF I$="SUR" THEN  PRINT "SU
R": GO TO 100
 220 IF I$="OESTE" THEN  PRINT "
OESTE": GO TO 100
 230 IF I$="ESTE" THEN  PRINT "E
STE": GO TO 100
 240 PRINT "PETICION NO VALIDA"
```

El empleo de variables enteras, una opción preferente

Con todo esto se logrará presentar de nuevo en la pantalla la palabra que se había puesto pero, en realidad, no conseguimos nada nuevo. Como modelo con el que trabajar, empezaremos en una posición definida por X = 0 e Y = 0 e indicaremos el movimiento con un más o un menos en relación a este punto. Obsérvese que se utilizan variables enteras siempre que es posible dado que su tratamiento es más rápido que el de los números reales. Con esto se elimina la posibilidad de conflicto con las variables reservadas.

```
10 LET X=0:LET Y=0
```

Ahora hemos de añadir la respuesta real al comando, así como el mensaje indicando que se ha comprendido (véase el Diagrama de flujo 2.3).

```
 200 IF I$="NORTE"THEN PRINT "NO
RTE":LET Y=Y-1:GO TO 100
 210 IF I$="SUR" THEN  PRINT "SU
R": LET Y=Y+1: GO TO 100
 220 IF I$="OESTE" THEN  PRINT "
OESTE": LET X=X-1: GO TO 100
 230 IF I$="ESTE" THEN  PRINT "E
STE": LET X=X+1: GO TO 100
```

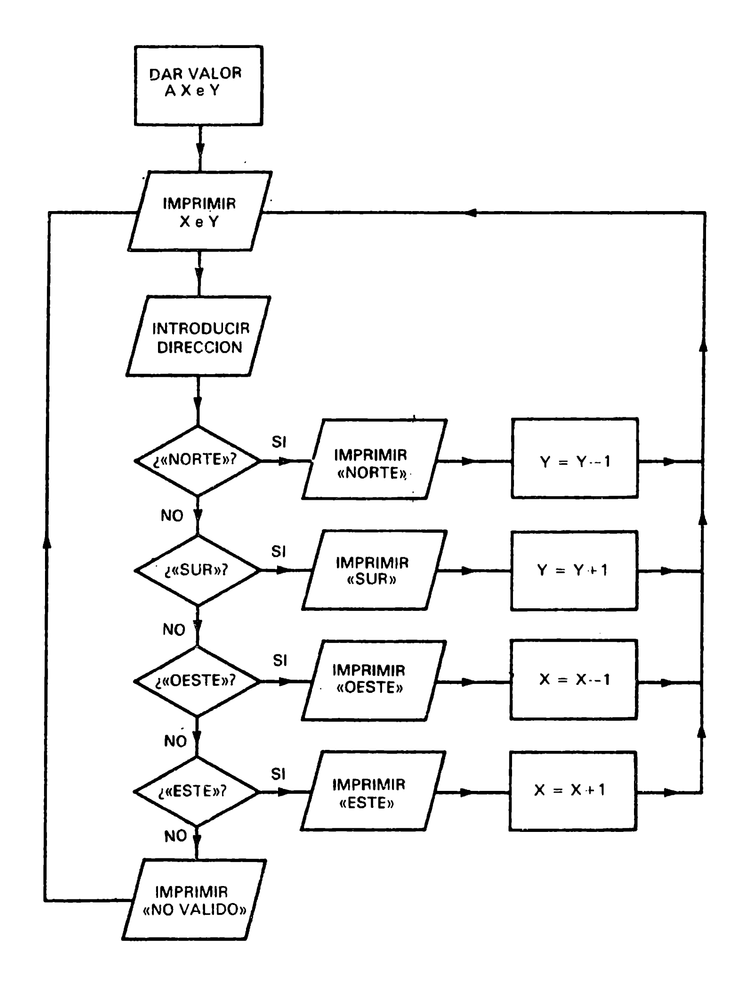


Diagrama de flujo 2.3. La incorporación de una respuesta

Esta modificación muestra realmente la posición adecuada con respecto al origen. Así, para ver qué está sucediendo y dónde se encuentra uno, se añadirá un mensaje con la posición del momento:

```
110>PRINT ,"X";X,"Y";Y
```

Utilización de subrutinas

Por supuesto que se trataba de un ejemplo muy sencillo pero cuando los resultados de la acción son más complicados, es mejor poner las respuestas en subrutinas.

```
 200 IF I$="NORTE"THEN GO SUB 20
00:LET Y=Y-1:GO TO 100
 210 IF I$="SUR" THEN  GO SUB 20
00: LET Y=Y+1: GO TO 100
 220 IF I$="OESTE" THEN  GO SUB
2000: LET X=X-1: GO TO 100
 230 IF I$="ESTE" THEN  GO SUB 2
000: LET X=X+1: GO TO 100
2000 PRINT "IR AL NORTE": LET Y=
Y-1: RETURN
2100 PRINT "IR AL SUR": LET Y=Y+
1: RETURN
2200 PRINT "IR AL OESTE": LET X=
X-1: RETURN
2300 PRINT "IR AL ESTE": LET X=X
+1: RETURN
```

La eficacia de las subrutinas de acción

Se puede extender la utilización de las instrucciones condicionales IF-THEN *ad infinitum* (o mejor, ¡ad memoriam finitum!) pero viene a ser una manera complicada de hacer cosas que crea problemas cuando se desea realizar programas más complejos. Una forma más elegante de hacerlo con comandos y respuestas, es introducirlos en instrucciones DATA y después leerlas, mediante la instrucción READ, cuando sea necesario. Si se ponen tales comandos y respuestas por parejas en las instrucciones DATA, es más difícil que se mezclen y resulta más fácil su lectura (véase la Tabla 2.1).

```
9000 DATA "NORTE","IR AL NORTE",
"SUR","IR AL SUR","OESTE","IR AL
 OESTE","ESTE","IR AL OESTE"
```

COMANDO (C$)	RESPUESTAS (R$)
NORTE	IR HACIA EL NORTE
SUR	IR HACIA EL SUR
OESTE	IR HACIA EL OESTE
ESTE	IR HACIA EL ESTE

Tabla 2.1. Comandos y respuestas

Todas las instrucciones condicionales IF-THEN pueden sustituirse ahora por un sencillo bucle que compara las entradas con cada comando, leyendo, mediante la instrucción READ, cada uno de estos elementos de la instrucción DATA correspondiente en la variable C$ (véase el Diagrama de flujo 2.4) y poniendo en pantalla la respuesta oportuna por medio de R$ si la comparación es positiva. Obsérvese que el «puntero» de los datos debe restaurarse (RESTORE) para que siempre se empiece a leer, mediante la instrucción READ, desde el primer elemento de la lista de datos.

```
200 RESTORE :FOR N=1TO 4
210 READ C$,R$
220 IF I$=C$ THEN  PRINT R$: GO
TO 100
230 NEXT N
```

Si la entrada ahora, I$, es igual a un determinado comando, el programa salta del bucle después de presentar en pantalla la respuesta adecuada (R$).

En realidad nos volvemos a encontrar como antes, sin haber hecho nada, por lo que necesitamos poder llamar estas subrutinas de acción. En primer lugar, preparemos la salida del bucle si la comparación es positiva para ir a una subrutina nueva en la línea 300.

```
220 IF I$=C$THEN PRINT R$:GO TO
300
```

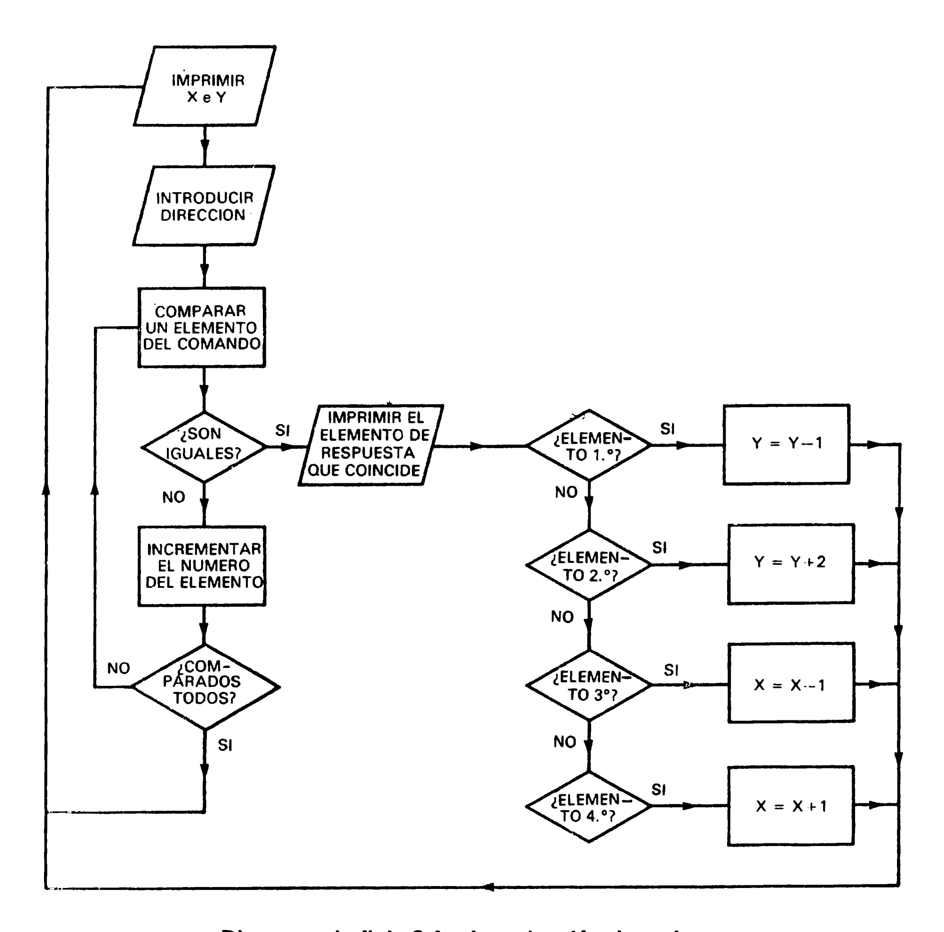


Diagrama de flujo 2.4. Introducción de mejoras

Todavía tenemos un «puntero» para indicar la palabra que se adapta al comando de entrada ya que N (el número de los elementos DATA comprobados) mantiene su valor. Podemos utilizar esto para acudir a las rutinas adecuadas que son similares a las que pusimos anteriormente con la diferencia de que no hay necesidad de definir el mensaje particular que ya se ha presentado con la variable alfanumérica R$.

```
300 IF N=1THEN GO SUB 2000
301 IF N=2 THEN  GO SUB 2100
```

```
 302 IF N=3 THEN  GO SUB 2200
 303 IF N=4 THEN  GO SUB 2300
 309 GO TO 100
2000 LET Y=Y-1: RETURN
2100 LET Y=Y+1: RETURN
2200 LET X=X-1: RETURN
2300 LET X=X+1: RETURN
```

La ampliación de vocabulario

La lista de datos se puede ampliar para contener más palabras. Por ejemplo, es posible añadir direcciones intermedias de la brújula que cambien los ejes X e Y.

```
9010 DATA "NORESTE","IR AL NORES
TE","SURESTE","IR AL SURESTE"
9020 DATA "SUROESTE","IR AL SURO
ESTE","NOROESTE","IR AL NOROESTE
"
```

y agregar algunas subrutinas más (téngase en cuenta que el bucle de exploración de los datos también debe incrementarse):

```
 200 RESTORE : FOR N=1 TO 8
 304 IF N=5 THEN  GO SUB 2400
 305 IF N=6 THEN  GO SUB 2500
 306 IF N=7 THEN  GO SUB 2600
 307 IF N=8 THEN  GO SUB 2700
2400 LET Y=Y-1: LET X=X+1: RETUR
N
2500 LET Y=Y+1: LET X=X+1: RETUR
N
2600 LET Y=Y+1: LET X=X-1: RETUR
N
2700 LET Y=Y-1: LET X=X-1: RETUR
N
```

Método para eliminar redundancias

Hasta ahora todas las respuestas han incluido la palabra «IR» que se ha puesto en cada elemento de la declaración DATA. No perjudica la práctica de escribir en el teclado pero sería mucho más fácil definir esta palabra común como una variable de cadena. Obsérvese que se incluye un espacio al final para separarla de la palabra siguiente.

```
40 LET G$="IR AL "
```

Se puede pues eliminar la palabra repetida y, en su lugar, poner G$ a continuación del comando.

```
 210 IF I$=C$THEN PRINT G$;R$:GO
 TO 300
9000 DATA "NORTE","NORTE","SUR",
"SUR","OESTE","OESTE","ESTE","ES
TE"
9010 DATA "NORESTE","NORESTE","S
URESTE","SURESTE"
9020 DATA "SUROESTE","SUROESTE",
"NOROESTE","NOROESTE"
```

Esto está empezando a parecer bastante tonto ya que ambas matrices (arrays) ahora contienen exactamente las mismas palabras. ¿Por qué no desechar la palabra respuesta, R$, y poner simplemente C$? Bien, en este caso, pudiera hacerse sin ningún problema pero cuando la contestación no es una simple repetición del comando de entrada (como ocurre con frecuencia), resulta esencial la segunda palabra o frase.

Si se examinan más cuidadosamente todas estas subrutinas, se apreciará que hacen solamente una cosa: actualizar los valores de X e Y. ¡Podríamos incluir esta información en los datos originales y desprendernos de ellos conjuntamente! Necesitamos añadir los valores apropiados de las coordenadas X e Y en las líneas de los datos después de cada respuesta y leer, mediante la instrucción READ, esta

información en bloques de cuatro (ENTRADA, RESPUESTA, MOVIMIENTO EN X, MOVIMIENTO EN Y). Véase la Tabla 2.2.

COMANDO C$	RESPUESTA R$	MOVIMIENTO EN X XU	MOVIMIENTO EN Y YU
NORTE	NORTE	0	—1
SUR	SUR	0	1
OESTE	OESTE	—1	0
ESTE	ESTE	1	0
NORESTE	NORESTE	1	—1
SURESTE	SURESTE	1	1
SUROESTE	SUROESTE	—1	1
NOROESTE	NOROESTE	—1	—1

Tabla 2.2. Movimientos en X e Y incorporados a los datos

```
9000 DATA "NORTE","NORTE",0,-1,"
SUR","SUR",0,1,"OESTE","OESTE",-
1,0
9010 DATA "NORESTE","NORESTE",1,
-1,"SURESTE","SURESTE",1,1
9020 DATA "SUROESTE","SUROESTE",
-1,1,"NOROESTE","NOROESTE",-1,-1
```

Ahora podemos borrar las líneas 300 a 2700 y modificar la 210 y 220 para que X e Y se actualicen aquí (véase el Diagrama de flujo 2.5).

```
 210 READ C$,R$,XU,YU
 220 IF I$=C$ THEN  PRINT G$;R$:
 LET X=X+XU: LET Y=Y+YU: GO TO 1
00
```

Este modelo general de poner toda la información de forma encadenada es un rasgo común que se utilizará varias veces en programas ulteriores de este libro.

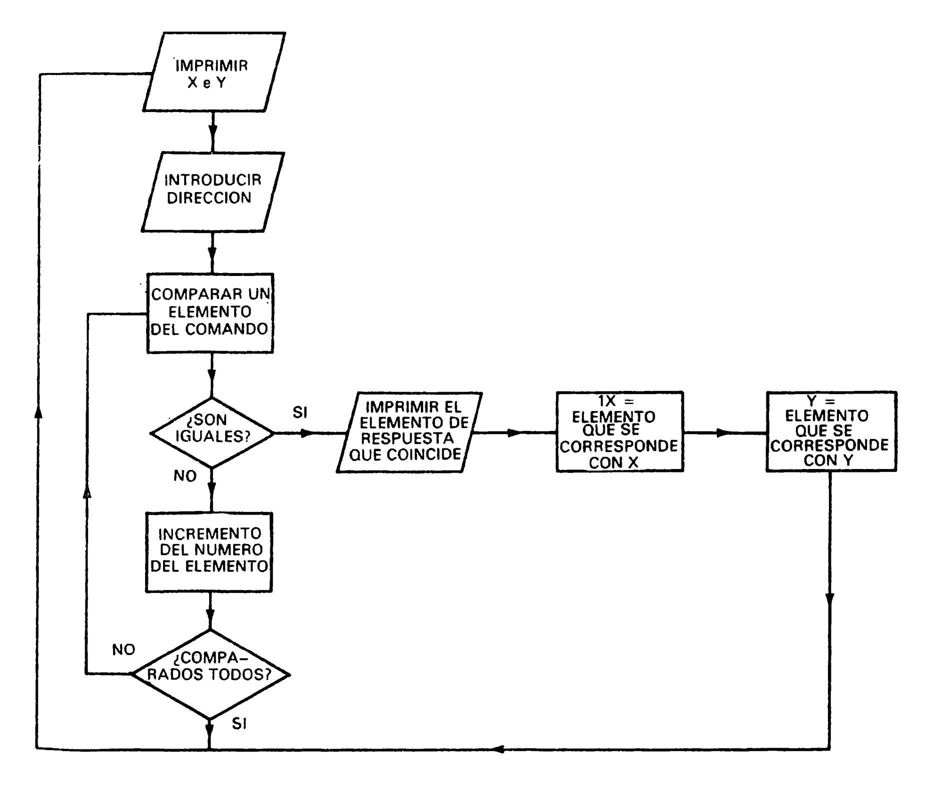


Diagrama de flujo 2.5. Utilización de datos enlazados

Comandos abreviados

Hasta ahora hemos utilizado palabras completas como comandos, pero esto significa que hay que escribir mucho para dar instrucciones al ordenador. Si resulta pesado, se puede pensar en cambiar las palabras enteras por su inicial y, en consecuencia, entrar mediante la instrucción INPUT, una sola letra. Sin embargo, a menos que se utilicen letras aleatorias, el problema irá bien hasta que tengamos dos palabras que empiecen por la misma letra. Para codificar las ocho direcciones de la brújula utilizadas anteriormente tendremos que hacer utilizar hasta dos de las letras siguientes: N, NE, E, SE, S, SO, O, NO.

```
9000 DATA "N","NORTE",0,-1,"S","
SUR",0,1,"O","OESTE",-1,0,"E","E
STE",1,0
9010 DATA "NE","NORESTE",1,-1
9020 DATA "SO","SUROESTE",-1,1
```

Obsérvese que sólo han cambiado los comandos. El ordenador da una descripción completa de la dirección ya que todavía estamos empleando aquella segunda parte de la lista de datos que contiene la respuesta.

La comparación de palabras y de claves abreviadas

En todos los programas anteriores nos hemos asegurado de que la palabra de entrada coincidía exactamente con la matriz que constituía el comando. Sin embargo, sería útil que pudieran ser aceptables ciertas palabras con el mismo significado. Por ejemplo, se podría comprobar si la primera letra de la palabra se adaptase a la clave abreviada comparando su primer carácter (tomando I$(1)).

```
190 LET I$=I$(1)
```

Sería válido con NORTE, SUR, ESTE Y OESTE, pero existen problemas obvios al tratar de las posiciones intermedias. Además, hay muchas palabras que empiezan con las letras N, S, E y O, que serían igualmente aceptables para el ordenador como dirección válida. Por ejemplo: NO NORTE nos enviaría precisamente en esa dirección. Un proceso más selectivo sería comparar varias letras en lugar de una. En este ejemplo las tres primeras letras de las cuatro direcciones principales son características.

NOR
SUR
EST
OES

Si se emplean esas palabras como comandos, resultarían igualmente aceptables:

NORTE
NORTEÑO
NORDICO

pero serían rechazadas otras como:

NOS
NOVIA
NOTA
NORIA

Todo lo que tenemos que hacer es tomar las tres primeras letras de la palabra de entrada, I$(TO 3), incluyéndose, naturalmente, la comprobación de que el número de letras de I$ no es inferior a dicho número.

```
 190 IF LEN (I$)<3 THEN  GO TO 1
00
 195 LET I$=I$( TO 3)
9000 DATA "NOR","NORTE",0,-1,"SU
R","SUR",0,1,"EST","ESTE",1,0
```

Comandos secuenciales

En las rutinas anteriores hemos considerado las posiciones intermedias de las direcciones de la brújula como entidades separadas pero si pudiéramos dar una secuencia de comandos, al mismo tiempo, no necesitaríamos hacer esto. Siempre hay más de un camino para llegar a un punto y si se pudiera comprender simultáneamente más de un comando, no nos tendríamos que preocupar en comprobar todas las direcciones tales como NORESTE, ya que podría tratarse por la combinación de NORTE y ESTE. Esto nos lleva a la significativa cuestión de cómo dividir una entrada en palabras. Primeramente tenemos que preguntarnos cómo reconocer que una serie de caracteres constituye una palabra separada. La respuesta es, por supuesto, la separación que vemos entre ellas. Si buscamos los espacios

podremos descomponer la entrada en palabras separadas que analizaremos individualmente. La forma más sencilla de buscar los espacios en BASIC es mediante la instrucción «INSTR» que analiza la totalidad de una cadena de búsqueda designada para ver si coincide con otra propuesta. Desgraciadamente el BASIC de Sinclair no dispone de ella por lo que tendremos que utilizar una serie de expresiones del BASIC que sustituyan tal instrucción. Dicha serie se pondrá en una subrutina en la línea 5000 a la que nos referiremos en lo que sigue del libro simplemente como la rutina INSTR.

```
5000 FOR N=1 TO LEN (I$)
5010 IF I$(N)=" " THEN  LET SP=N
: RETURN
5020 NEXT N
5030 LET SP=0
5040 RETURN
```

Esta rutina comprobará si el primer carácter de I$ es un espacio. Si no lo es, seguirá haciendo la comprobación automáticamente, hasta alcanzar el final de la citada variable alfanumérica I$. Si no se encuentra ningún espacio, la variable SP se pondrá a cero y, si se encuentra alguno, el valor de dicha variable SP será el número de caracteres de I$ donde se halla el espacio (véase el Diagrama de flujo 2.6).

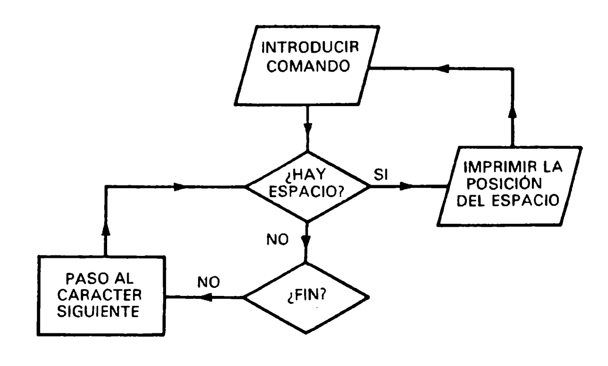


Diagrama de flujo 2.6. Búsqueda de espacios

Se necesita llamar a esta rutina desde la principal e imprimiremos el resultado cuando se produzca el RETURN, de forma que veamos lo que está sucediendo.

```
130 GO SUB 5000
140 PRINT SP: GO TO 100
```

Pruébese la subrutina con:

NOR OES

SP 4

NORO ESTE

SP 5

NOR NOR OESTE

SP 4

Obsérvese que SP nos da la longitud de la palabra incluyendo el espacio pero sólo halla el primero de éstos. Para encontrarlos todos hay que trabajar más. En primer lugar, bórrese la línea 140.

Analicemos lógicamente la entrada desde el principio (extremo izquierdo). Substituyamos I$(TO 3) por I$(ST TO ST + 2) para que podamos observar cualquier combinación de tres letras del total de I$. Para hacerlo más evidente llamaremos al resultado de esto —W$—, ya que nos muestra la posición de una palabra. Para empezar debemos iniciar la investigación haciendo ST igual a 1 y añadir un espacio delante de I$ para que también se encuentre la primera palabra (véase el Diagrama de flujo 2.7).

```
125 LET ST=1: LET I$=" "+I$
130 GO SUB 5000
190 LET W$=I$(ST TO ST+2)
210 IF W$=C$ THEN  PRINT G$;R$:
```

```
LET X=X+XU: LET Y=Y+YU: GO TO 1
00
5000 FOR N=ST TO LEN (I$)
```

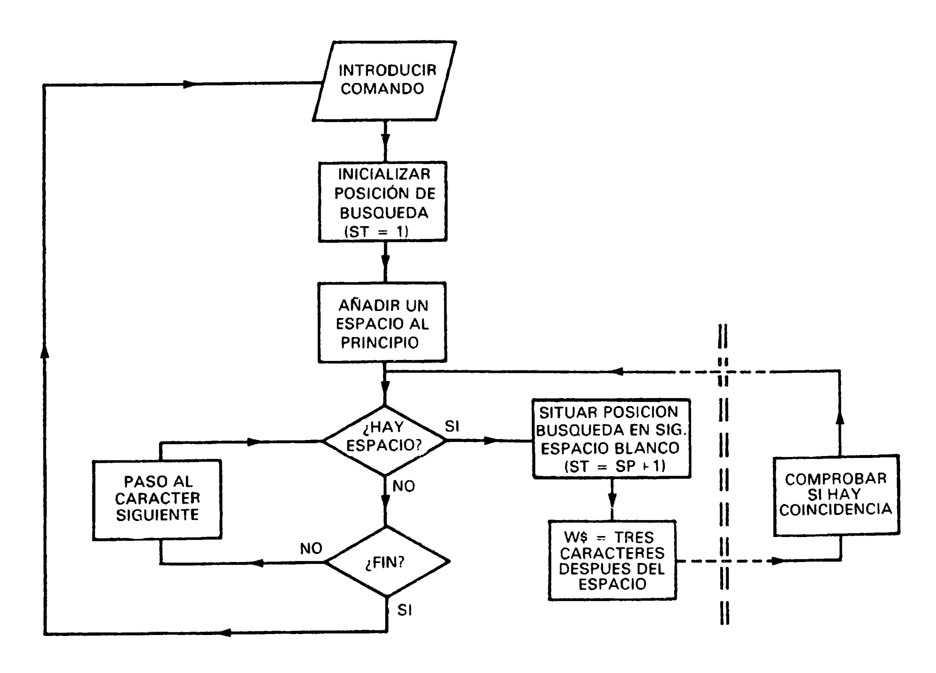


Diagrama de flujo 2.7. Búsqueda de una palabra clave

Si ejecutamos esto tal como está, seguiremos encontrando todavía la primera palabra ya que tenemos el «GO TO 100» al final de la línea 210. Sin embargo, enviando simplemente otra vez el programa a la comprobación INSTR, en la línea 130, tampoco nos ayuda ya que empezará siempre la comprobación desde el principio de I$ y volverá a encontrar el mismo primer espacio. Una vez lo hayamos encontrado, tenemos que desplazar la posición inicial —ST—, para la siguiente búsqueda sobre el carácter que sigue a dicho espacio, SP + 1. Cuando ya no pueden encontrarse más espacios, se ha alcanzado el final de la expresión de entrada y podemos remitirnos nuevamente a la instrucción GO TO 100.

```
 140 IF SP>0 THEN  LET ST=SP+1:
GO TO 190
 150 GO TO 100
 210 IF W$=C$ THEN  PRINT G$;R$:
 LET X=X+XU: LET Y=Y+YU: GO TO 1
30
```

Ahora la entrada:

NOR OESTE

produce:

IR NORTE
IR OESTE

y NOR NOR ESTE

se decodifica en:

IR NORTE
IR NORTE
IR ESTE

La ejecución del bucle de comparación palabra propuesta / palabra almacenada

Quedaría mucho mejor si se suprimieran los «IR» redundantes, y se pusieran las direcciones en la misma línea. Necesitamos la instrucción PRINT G$ una vez, inmediatamente delante de la comprobación INSTR. Ahora cada vez que se ejecuta el bucle de comparación de la palabra propuesta con las almacenadas, se presentará en pantalla R$ (PRINT), si tal comparación es positiva. Como existe un punto y coma después de R$, las palabras se presentarán en la misma línea aunque todavía necesitamos añadir espacios entre ellas. Finalmente agregamos una simple instrucción PRINT antes de volver a la nueva entrada para desplazar la posición del cursor a la línea siguiente.

```
126 PRINT G$;
145 PRINT
210 IF W$=C$ THEN  PRINT R$;" "
;: LET X=X+XU: LET Y=Y+YU: GO TO
130
```

Ahora:

NOR ESTE SUR OESTE

nos enviaría en círculos con el mensaje:

IR NORTE ESTE SUR OESTE

3

Estudio del lenguaje natural

Hasta ahora sólo nos hemos comunicado con el ordenador de una forma muy restringida ya que únicamente ha sido programado para comprender muy pocas palabras o letras que reconoce exclusivamente cuando se introducen en una forma precisa. Por ejemplo, si se pone un espacio delante o detrás de la expresión o palabra INPUT (entrada), el ordenador la rechazará ya que lo que hace es comprobar si dos cadenas son exactamente iguales.

Por otra parte, en el mundo real todos utilizamos lo que se conoce por «lenguaje natural» que es una cosa muy complicada y extremadamente variable únicamente interpretado con eficacia por el cerebro humano. Incluso si olvidamos, por un momento, las diferencias entre las variaciones regionales, el proceso del lenguaje produce un infinito número de problemas.

Ni los más complejos sistemas del mundo pueden pretender resolver todos los problemas. Hay una vieja historia que ilustra esta situación muy bien. La CIA desarrolló un soberbio programa de traducción del inglés al ruso y viceversa. Confiando en impresionar al Presidente de los Estados Unidos, se preparó una demostración de sus posibilidades en la que cuanto decía se pasaba al ruso y después esta versión se traducía nuevamente al inglés. Quedó admirado y totalmente absorto hasta que uno de sus ayudantes le recordó que había olvidado que la Primera Dama estaba esperándole fuera. El Presidente comentó: «Ojos que no ven, corazón que no siente». Su sorpresa fue grande cuando oyó a la máquina traducir: «Ciego insensible».

El tratamiento de las oraciones gramaticales, por ordenador

Es de todos conocido que el lenguaje real se compone de oraciones, pero ¿qué es realmente una oración? Bien, la forma más elemental de reconocer una es mediante el punto que lleva al final. Sin embargo, si vamos a tratar con ellas habremos de profundizar mucho más.

El «Oxford Dictionary» define una oración en estos términos: «Una serie de palabras encadenadas oralmente o por escrito que forman una expresión gramatical completa con un solo sentido, conteniendo generalmente sujeto y predicado, haciendo una declaración, una pregunta, una orden o una petición». También admite en un sentido más libre «parte de un escrito o expresión oral comprendida entre dos detenciones (puntos)». ¿Se puede hacer esto más comprensible? Las complicaciones y los desvíos frecuentes de la lógica en un idioma son enormes. ¿Cómo podemos esperar que un ordenador pueda con ello?

Bien, empecemos viendo algunos ejemplos de oraciones sencillas.

YO QUIERO

Tenemos un sujeto —YO— y un verbo —QUIERO—.

YO QUIERO BIZCOCHOS

Aparece un objeto: BIZCOCHOS

YO QUIERO BIZCOCHOS MANTECOSOS

Se califica el objeto con el determinante MANTECOSOS.

YO SIEMPRE QUIERO BIZCOCHOS MANTECOSOS

Se califica el verbo con el determinante adverbial SIEMPRE.

La palabra más importante de los ejemplos anteriores es QUIERO ya que comprende la idea principal. El segundo ejemplo es más informativo pues nos dirige a un objeto en particular: los BIZCO-

CHOS. La adición del adjetivo MANTECOSOS nos da más datos sobre el objeto deseado, pero la cosa se complica nuevamente cuando se incluye el adverbio SIEMPRE.

¿Cómo puede analizar tales frases un ordenador? La respuesta es quizás encontrar alguna estructura lógica en la oración. ¿Qué reglas pueden establecerse para este ejemplo?

1) Todo empieza con un sujeto —YO— y termina con un punto final.
2) Las palabras que siguen al verbo son el complemento directo (pues empleamos un verbo transitivo).
3) La palabra delante del complemento es el verbo.
4) Cuando al verbo no precede el pronombre sujeto, se trata de un adverbio.
5) El adjetivo debe seguir al objeto.

Un programa para analizar frases

Preparemos un programa que analice las frases que se aplican al ordenador. Naturalmente que con las limitaciones que nos establecen las reglas anteriores.

Para empezar necesitamos un vocabulario de objetos, adjetivos y adverbios con que trabajar, que habrán de ponerse en instrucciones DATA agrupados según la clase. Los primeros elementos son seis objetos, seguidos de seis adjetivos y por último tres adverbios.

```
9000 REM COMPLEMENTOS OBJETO
9010 DATA "BIZCOCHOS","BOLLOS","
PASTELES"
9020 DATA "CAFE","TILA","PAN"
9030 REM ADJETIVOS
9040 DATA "CRUJIENTES","TIERNOS"
,"MANTECOSOS"
9050 DATA "FRIO","CALIENTE","TEM
PLADO"
9060 REM ADVERBIOS
9070 DATA "SIEMPRE","NUNCA","NOR
MALMENTE"
```

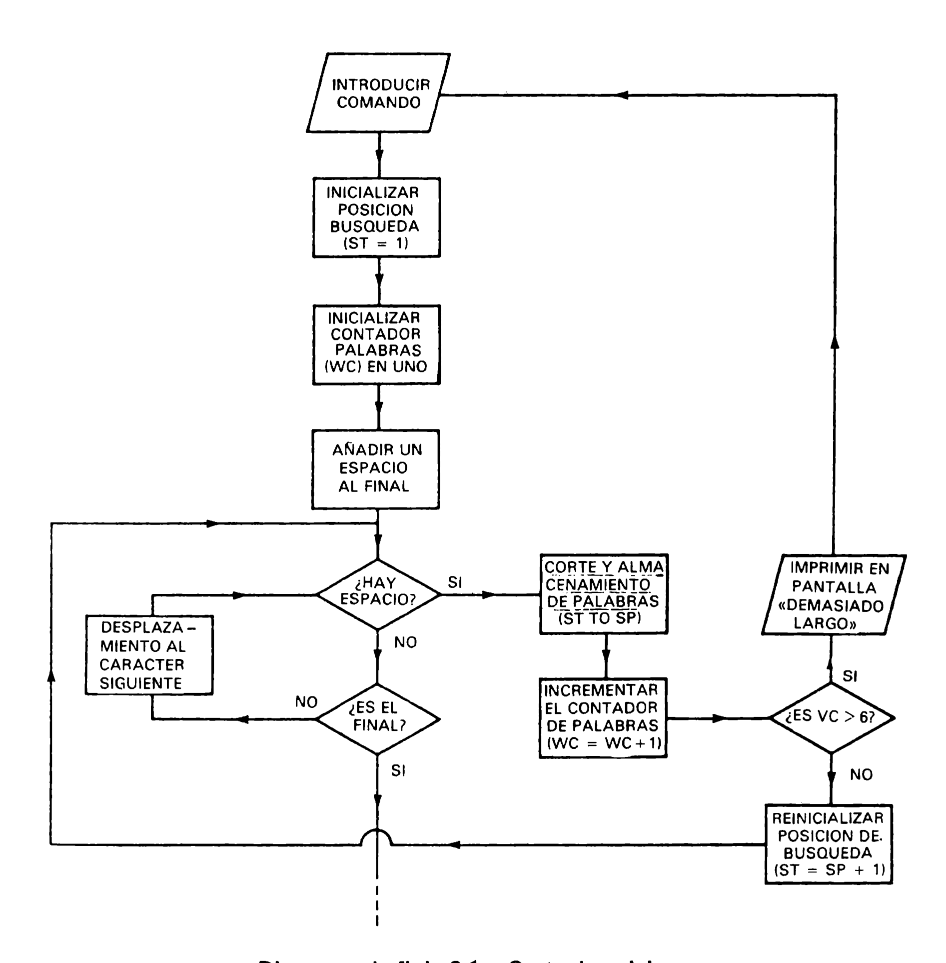


Diagrama de flujo 3.1. Corte de palabras

Ahora necesitamos descomponer la oración en palabras (véase el Diagrama de flujo 3.1). Nuevamente haremos esto por medio de la instrucción INSTR (véase Capítulo 2) que busca los espacios. Para hacer las cosas más fáciles añadiremos un espacio al final de I$ con lo que igualaremos el formato de todas las palabras.

```
100 INPUT I$: PRINT I$
110 LET ST=1
120 LET I$=I$+" "
```

```
 130 GO SUB 5000
 190 GO TO 130
5000 FOR N=ST TO LEN (I$)
5010 IF I$(N)=" " THEN  LET SP=N
: RETURN
5020 NEXT N
5030 LET SP=0
5040 RETURN
```

Se llega al final de la oración cuando no pueden encontrarse más espacios.

```
140 IF SP=0 THEN  GO TO 200
```

Si se encuentra un espacio se corta la parte de I$ desde ST (principio del análisis) hasta SP (el espacio, lo que comprende la longitud total de la palabra) y se guarda en una cadena de palabras: W$(WC).

```
 10 DIM W$(10,12)
150 LET W$(WC)=I$(ST TO SP)
```

Empezamos con ST = 1 para que el análisis comience en el primer carácter de la cadena de entrada. La variable contadora de palabras WC se pone en uno para que la primera hallada se guarde como primer elemento de esta cadena de almacenamiento.

```
110 LET ST=1: LET WC=1
```

Se incrementa el contador de palabras (para que el siguiente elemento de la cadena W$ pueda admitir la próxima voz). También se comprueba que no haya más de cinco palabras en la oración. La posición de inicio para la siguiente búsqueda se establece en una unidad más que la del último espacio y se continúa la investigación.

```
 160 LET WC=WC+1
 170 IF WC>6 THEN  PRINT "FRASE
DEMASIADO LARGA": GO TO 100
 180 LET ST=SP+1
```

Veamos cómo el programa nos descompone ahora la frase en palabras, las presenta y nos dice el número de ellas. Ejecútese el listado que sigue que es la concatenación de los precedentes con las siguientes modificaciones: en la línea 150 se ha añadido un PRINT para la presentación de las palabras, la 185 para salir del bucle al terminar el contaje y la 5045 para presentar en pantalla el número de palabras de la frase. (Obsérvese que en esta línea se pone WC-1 porque se añadió un espacio a la cadena I$, que se contaría en otro caso como palabra).

```
  10 DIM W$(5,12)
 100 INPUT "PONGA SU FRASE";I$:
PRINT I$
 110 LET ST=1: LET WC=1
 120 LET I$=I$+" "
 130 GO SUB 5000
 140 IF SP=0 THEN  GO TO 200
 150 LET W$(WC)=I$(ST TO SP): PR
INT ''W$(WC)
 160 LET WC=WC+1
 170 IF WC>6 THEN  PRINT "FRASE
DEMASIADO LARGA": GO TO 100
 180 LET ST=SP+1
 185 IF ST>=LEN I$ THEN  GO TO 5
045
 190 GO TO 130
5000 FOR N=ST TO LEN I$
5010 IF I$(N)=" " THEN  LET SP=N
: RETURN
5020 NEXT N
5030 LET SP=0
5040 RETURN
5045 PRINT ''"NUMERO DE PALABRAS
 EN LA FRASE= ";WC-1
```

```
9000 REM OBJETOS
9010 DATA "BIZCOCHOS","BOLLOS","
PASTELES"
9020 DATA "CAFE","TILA","PAN"
9030 REM ADJETIVOS
9040 DATA "CRUJIENTES","TIERNOS"
,"MANTECOSOS"
9050 DATA "FRIO","CALIENTE","TEM
PLADO"
9060 REM ADVERBIOS
9070 DATA "SIEMPRE","NUNCA","NOR
MALMENTE"
```

Continuemos ahora con una prueba para ver si hay identidad entre las palabras objeto de la frase y las correspondientes contenidas en la instrucción DATA O$ que contiene el vocabulario (véase el Diagrama de flujo 3.2).

Dado que el BASIC de Sinclair utiliza cadenas de longitud fija de acuerdo con el dimensionado que establece la instrucción DIM W$ (10,12), hay que añadir el adecuado número de espacios al final de la palabra objeto guardada en la instrucción DATA para poder compararla con la correspondiente palabra entrada. Logramos esto agregando doce espacios en el extremo derecho de O$ y cortando después los doce primeros caracteres de la cadena resultante.

```
205 READ O$: LET O$=(O$+"
     ")( TO 12)
```

Sólo se comprueban las palabras 3 y 4 ya que son las únicas posibles para el objeto en nuestro restringido formato de la frase. Se pasa por dos rutinas diferentes según la posición de la palabra correspondiente de la frase. Si no hay identidad se presenta el oportuno mensaje en pantalla y se solicita una nueva entrada. La instrucción RESTORE al principio de la línea 200 lleva siempre el puntero de datos a su origen.

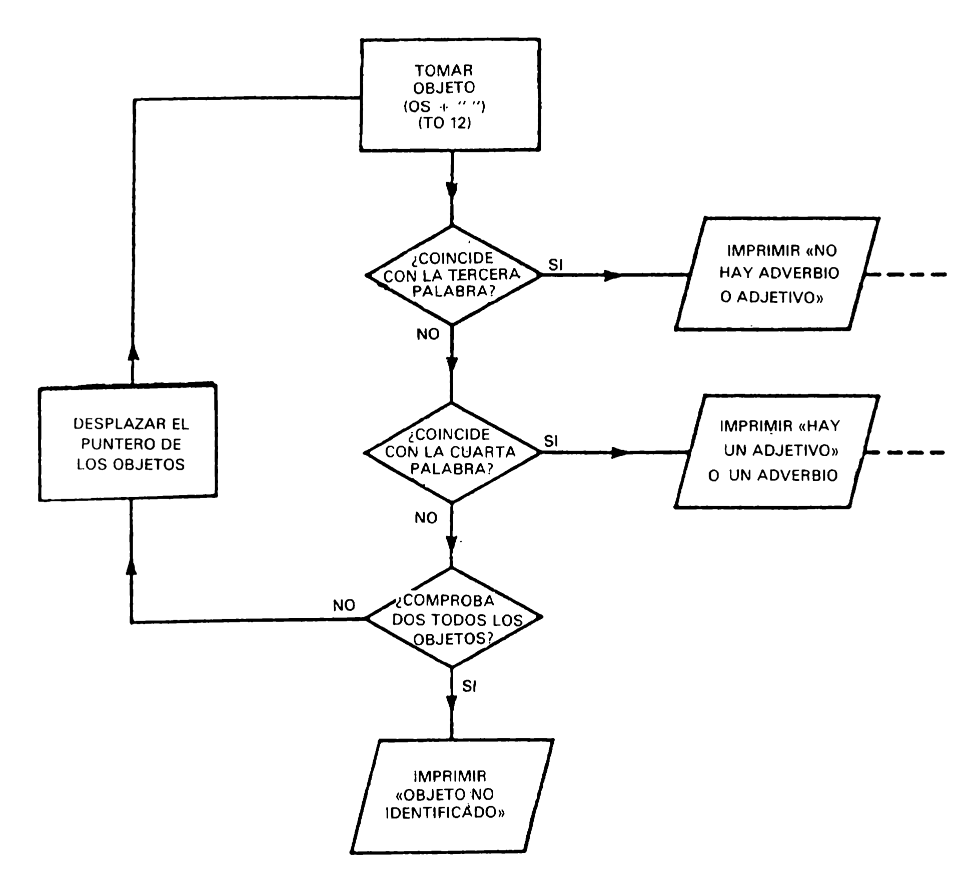


Diagrama de flujo 3.2. Buscando la coincidencia

```
 200 RESTORE : FOR N=1 TO 6
 210 IF W$(3)=O$ THEN  GO TO 500
 220 IF W$(4)=O$ THEN  GO TO 600
 240 NEXT N
 250 PRINT '"OBJETO NO IDENTIFIC
ADO"
 260 PRINT '"PONGA UN OBJETO DEL
 VOCABULARIO"
 265 GO TO 100
```

Si se encontró el objeto como palabra tres no puede haber adverbio (que ha de estar después del verbo).

```
 500 PRINT '"NO HAY ADVERBIO EN
LA FRASE"
 510 GO TO 640
```

Si el objeto se encontró en la palabra cuatro, tiene que haber un adverbio en la frase (véase el Diagrama de flujo 3.3).

```
600 PRINT '"HAY UN ADVERBIO"
```

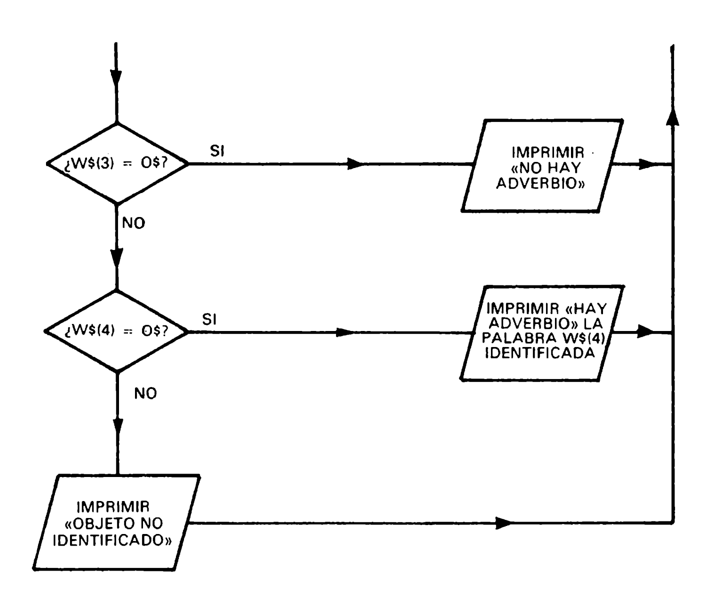


Diagrama de flujo 3.3. Coincidencia del adverbio y del objeto

Primero buscamos la identidad entre la segunda palabra y los adverbios contenidos en la instrucción DATA. Estos son leídos secuencialmente en la variable V$. Debemos recordar que el puntero de datos debe ponerse en el primer adverbio por medio de la instrucción RESTORE 9060 y V$ ha de completarse con espacios que permitan la comparación.

```
 610 RESTORE 9060: FOR N=1 TO 3
 615 READ V$: LET V$=(V$+"
      ")( TO 12)
 620 IF W$(2)=V$ THEN  GO TO 900
 630 NEXT N
 635 PRINT '"ADVERBIO ";W$(2);"N
O IDENTIFICADO"
```

Si hay identidad o coincidencia, la línea 620 nos remite a la 900 que presenta la palabra. Si no hay identidad, se presenta el correspondiente mensaje de no haber coincidencia en el caso del adverbio, diciéndose la palabra que no lo ha sido. A continuación se busca la identidad del adjetivo en la lista correspondiente que se asigna a la variable J$.

```
 640 RESTORE 9030: FOR N=1 TO 6
 645 READ J$: LET J$=(J$+"
      ")( TO 12)
 650 IF W$(5)=J$ THEN  GO TO 100
0
 652 IF W$(4)=J$ THEN  GO TO 100
5
 655 NEXT N
 670 IF WC<6 THEN  PRINT '"NO HA
Y ADJETIVO EN LA FRASE": GO TO 1
00
```

El adjetivo puede estar en la palabra 4 si no hay adverbio o en la 5 si lo hay, por lo que se usan dos rutinas, líneas 650 y 652, en las que si se cumple la condición nos remiten a las 1000 y 1005, una de las cuales nos identifica el adjetivo.

Resumamos ahora lo expuesto con un listado completo que nos analiza las frases que le pongamos, diciéndonos el adverbio y el adjetivo si los hay y cuáles son las palabras no identificadas en cada caso. Las frases han de tener un sujeto, un verbo y un adverbio o un adjetivo, o ambos, pero en el orden de las reglas que establecimos. La frase no puede tener más de cinco palabras ni me-

nos de cuatro. El vocabulario del programa es muy limitado pero nada impide que se extienda, si así se desea.

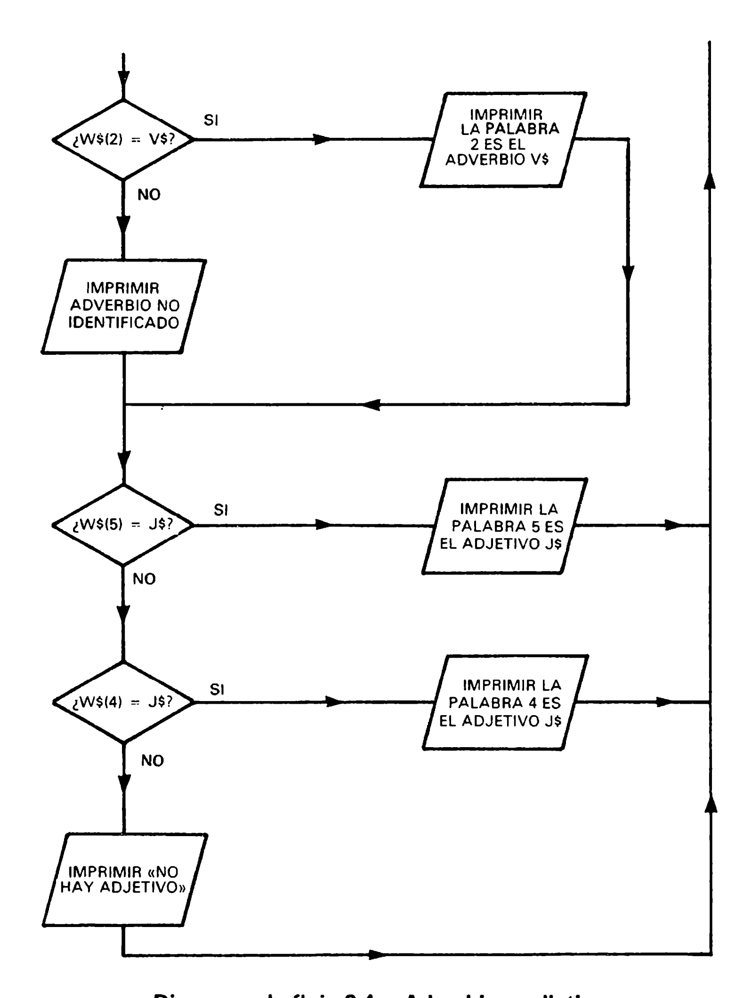


Diagrama de flujo 3.4. Adverbio y adjetivo

```
  90 REM PROGRAMA RESUMEN DE LO
TRATADO HASTA AHORA EN ESTE CAPI
TULO.
CONDICIONES QUE DEBEN DE SATISFA
CER LAS ORACIONES PARA LA ACTUAC
```

```
ION CORRECTA DEL PROGRAMA:
1.-SINTAXIS.PRONOMBRE,ADVERBIO,
VERBO,COMPLEMENTO OBJETO Y ADJET
IVO.
2.-PUEDE FALTAR EL ADVERBIO O EL
 ADJETIVO,PERO NO LOS DOS
  95 PRINT
  97 PRINT
 100 INPUT "PONGA SU FRASE ";I$:
 PRINT I$
 101 PRINT
 102 DIM W$(10,12)
 104 IF I$="ALTO" THEN  STOP
 110 LET ST=1: LET WC=1
 120 LET I$=I$+" "
 130 GO SUB 5000
 140 IF SP=0 THEN  GO TO 191
 150 LET W$(WC)=I$(ST TO SP): PR
INT W$(WC)
 160 LET WC=WC+1
 180 LET ST=SP+1
 190 GO TO 130
 191 PRINT '"NUM.DE PALABRAS: ";
WC-1
 192 IF WC>6 THEN  PRINT '"FRASE
 DEMASIADO LARGA.  (PULSAR UNA T
ECLA PARA SEGUIR)": PAUSE 0: CLS
 : CLEAR : GO TO 100
 195 IF WC<5 THEN  PRINT '"ORACI
ON INCOMPLETA. (PULSAR     UNA T
ECLA PARA SEGUIR)": PAUSE 0: CLS
 : CLEAR : GO TO 100
 200 RESTORE : FOR N=1 TO 6
 205 READ O$: LET O$=(O$+"
      ")( TO 12)
 210 IF W$(3)=O$ THEN  GO TO 500
 220 IF W$(4)=O$ THEN  GO TO 600
 240 NEXT N
 250 PRINT '"OBJETO NO IDENTIFIC
ADO"
```

```
 260 PRINT '"PONGA UN OBJETO DEL
 VOCABULARIO"
 270 PRINT '"(PULSE UNA TECLA PA
RA SEGUIR)"; PAUSE 0: CLS : GO T
O 100
 500 PRINT ''"NO HAY ADVERBIO EN
 LA FRASE"
 510 GO TO 640
 600 PRINT '"HAY UN ADVERBIO"
 605 PRINT '"IDENTIFICADA PALABR
A ";W$(4)
 610 RESTORE 9060: FOR N=1 TO 3
 615 READ V$: LET V$=(V$+"
      ")( TO 12)
 620 IF W$(2)=V$ THEN  GO TO 900
 630 NEXT N
 635 PRINT '"NO IDENTIFICADO ADV
ERBIO ";W$(2): GO TO 640
 640 RESTORE 9030: FOR N=1 TO 6
 645 READ J$: LET J$=(J$+"
      ")( TO 12)
 650 IF W$(5)=J$ THEN  GO TO 100
0
 652 IF W$(4)=J$ THEN  GO TO 100
5
 655 NEXT N
 670 IF WC<6 THEN  PRINT '"NO HA
Y ADJETIVO EN LA FRASE": GO TO 9
10
 680 PRINT '"ADJETIVO NO IDENTIF
ICADO.        (PULSAR UNA TECLA P
ARA SEGUIR)": PAUSE 0: CLS : GO
TO 100
 900 PRINT "LA PALABRA 2 ES EL A
DVERBIO:",,V$
 905 IF WC<6 THEN  PRINT '"NO HA
Y ADJETIVO EN LA FRASE": GO TO 9
10
 907 IF WC>=6 THEN  PRINT '"TAMB
IEN HAY UN ADJETIVO": GO TO 640
```

```
 910 PRINT '"(PULSAR UNA TECLA P
ARA SEGUIR)": PAUSE 0: CLS : GO
TO 100
1000 PRINT "LA PALABRA 5 ES EL A
DJETIVO:",,W$(5),"(PULSAR UNA TE
CLA PARA SEGUIR)": PAUSE 0: CLS
: GO TO 100
1005 PRINT '"LA PALABRA 4 ES EL
ADJETIVO:",,W$(4)
1010 PRINT '"(PULSAR UNA TECLA P
ARA SEGUIR)": PAUSE 0: CLS : GO
TO 100
5000 FOR N=ST TO LEN I$
5010 IF I$(N)=" " THEN  LET SP=N
: RETURN
5020 NEXT N
5030 LET SP=0
5040 RETURN
9000 REM OBJETOS
9010 DATA "BIZCOCHOS","BOLLOS","
PASTELES"
9020 DATA "CAFE","TILA","PAN"
9030 REM ADJETIVOS
9040 DATA "CRUJIENTES","TIERNOS"
,"MANTECOSOS"
9050 DATA "FRIO","CALIENTE","TEM
PLADO"
9060 REM ADVERBIOS
9070 DATA "SIEMPRE","NUNCA","NOR
MALMENTE"
```

Presentación en pantalla al ejecutarse el programa

```
YO NORMALMENTE COMPRO BIZCOCHOS
MANTECOSOS

YO
NORMALMENTE
```

```
COMPRO
BIZCOCHOS
MANTECOSOS

NUM.DE PALABRAS: 5

HAY UN ADVERBIO

IDENTIFICADA PALABRA BIZCOCHOS

LA PALABRA 2 ES EL ADVERBIO:
                   NORMALMENTE

TAMBIEN HAY UN ADJETIVO
LA PALABRA 5 ES EL ADJETIVO:
                   MANTECOSOS
<PULSAR UNA TECLA PARA SEGUIR>
```

La puntuación interpretada por el ordenador

Como ya hemos dicho anteriormente, se reconoce generalmente el final de una frase porque encontramos un punto aunque cuando se teclea el ordenador nos olvidamos frecuentemente de estas trivialidades. Pero, ¿qué ocurrirá en el programa que hemos preparado hasta ahora si un usuario «culto» introduce una correcta puntuación? Si lo pensamos, por un momento, veremos que el ordenador empezará a quejarse ya que no reconocerá la última palabra que será leída con el punto incorporado.

Necesitamos, por consiguiente, comprobar si el último carácter de la cadena de entrada I$ es un punto. El mejor lugar para hacerlo parece ser inmediatamente después de la entrada (INPUT). Si el último carácter es «.» basta rechazarlo y continuar como antes.

Añadiremos esto como una subrutina a la que se pasa tan pronto como se hace una entrada. Otros signos de puntuación pueden aparecer también al final de la frase por lo que leeremos el último carácter como perteneciente a una variable L$ que usaremos posteriormente (véase el Diagrama de flujo 3.5).

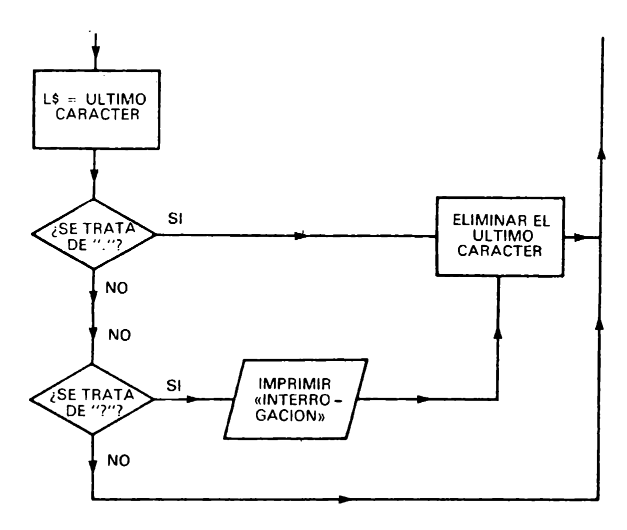


Diagrama de flujo 3.5. El tratamiento de la puntuación

```
 105 GO SUB 2000
2000 LET L$=I$(LEN (I$))
2010 IF L$="." THEN  GO TO 2100
2090 RETURN
2100 LET I$=I$( TO LEN (I$)-1):
RETURN
```

Una terminación frecuente de frase es el signo final de interrogación que indica el sentido de la oración. Podemos identificarlo de la misma manera y, de momento, nos limitaremos a informar de su presencia.

```
2030 IF L$="?" THEN  PRINT "INTE
RROGACION": GO TO 2100
```

La instrucción normal INPUT no aceptará nada detrás de una coma que se lee como terminación de datos. Sin embargo, INPUT LINE aceptará cualquier texto incluyendo las comas. (El único inconveniente de utilizar INPUT LINE es que se puede quedar atrapado en un bucle sin fin. A diferencia del INPUT normal, no se puede borrar ni detener el programa con el INPUT LINE).

```
100 INPUT LINE I$:PRINT I$
```

Las comas son útiles para indicar diferentes partes de una oración que pueden examinarse como sub-unidades. Sin embargo, en los casos sencillos es mejor eliminarlas y substituirlas por espacios antes de que la oración se descomponga en palabras (véase Diagrama de flujo 3.6). Obsérvese que esto sólo funcionará correctamente si no hay espacios detrás de la coma, ya que cualesquiera que sigan se interpretarán como una nueva palabra.

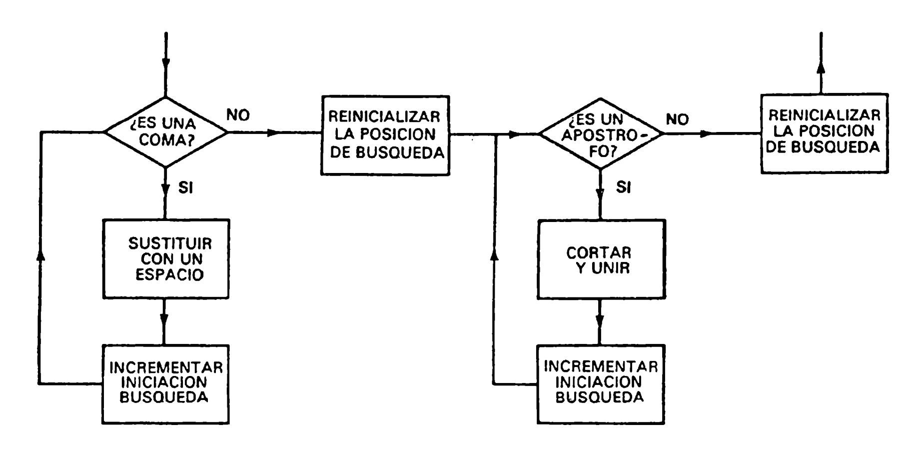


Diagrama de flujo 3.6. **Sustitución de comas y apóstrofos**

Vamos a modificar nuestra subrutina INSTR (véase Capítulo 2) para que podamos comprobar la aparición de I$ dentro de una cadena determinada (T$). Para clarificar las cosas a largo plazo, haremos que la variable señale la posición de la correspondencia o identidad en la cadena IS que puede intercambiarse después con cualquier número de variables diferentes como SP. Primero modificamos nuestra comprobación de espacios en el nuevo formato.

```
 130 LET T$=" ": GO SUB 5000: LE
T SP=IS
5010 IF I$(N)=T$ THEN  LET IS=N:
 RETURN
5030 LET IS=0
```

Ahora el mismo método puede utilizarse para buscar una coma, antes de substituirla con un espacio.

```
 115 LET T$=",": GO SUB 3000
3000 GO SUB 5000: LET CM=IS
3010 IF CM=0 THEN  LET ST=1: RET
URN
3020 LET I$=I$( TO CM-1)+" "+I$(
CM+1 TO )
3030 LET ST=CM+1
3040 GO TO 3000
```

Si añadimos esta línea se verá cómo desaparecen los signos de puntuación de la cadena, uno por uno.

```
3025 PRINT I$
```

Los apóstrofos pueden tratarse de la misma manera excepto que no los sustituimos con un espacio sino que simplemente juntamos las palabras.

```
 115 LET T$=",": GO SUB 3000: LE
T T$="'": GO SUB 3100
3100 GO SUB 5000: LET AP=IS
3110 IF AP=0 THEN  LET ST=1: RET
URN
3120 LET I$=I$( TO AP-1)+I$(AP+1
 TO )
3125 PRINT I$
3130 LET ST=AP+1
3140 GO TO 3100
```

Un método de análisis corrido

Aunque el método de analizar una frase descrito anteriormente es aprovechable, tiene la desventaja de que requiere que la frase

entrada tenga una forma particular restringida. Por ejemplo, si se aplica:

PASTELES TIERNOS ES LO QUE QUIERO

el ordenador responderá:

OBJETO NO ENCONTRADO

ya que sólo establecimos que los buscase en la tercera o cuarta palabra.

Haciendo un análisis corrido de toda la frase para buscar cada palabra clave sin tener que descomponer aquella previamente, se logra la ventaja de que permite una forma completamente libre. Con este método tomamos la primera palabra y tratamos de identificarla con el mismo número de letras de la cadena I$ empezando por la primera. Si esto falla, automáticamente repetimos la prueba empezando por el segundo carácter y así seguimos hasta entontrar la palabra o llegar al final de la cadena I$. Por ejemplo, si I$ es YO QUIERO PASTEL y la primera palabra del repertorio es «PASTEL», la comparación sería:

Paso 1	YO QUI	
Paso 2	O QUIE	
Paso 3	QUIER	
Paso 4	QUIERO	
Paso 5	UIERO	
Paso 6	IERO P	
Paso 7	ERO PA	
Paso 8	RO PAS	
Paso 9	O PAST	
Paso 10	PASTE	
Paso 11	PASTEL	(palabra encontrada)

Hasta ahora nuestra rutina INSTR sólo ha tratado de identificar un solo carácter pero tendremos que modificar la línea 5010 nuevamente para que tenga en cuenta la longitud de la palabra analizada. Utilizaremos una variable que acoja dicha longitud: WL.

```
5000 FOR N=1 TO LEN (I$)-WL+1
5010 IF I$(N TO (N+WL-1))=T$ THE
N  LET IS=N: RETURN
5020 NEXT N
5030 LET IS=0
5040 RETURN
```

Bórrese todo excepto las líneas de datos (DATA) (de la 9000 en adelante) y agréguense éstas para la identificación de un objeto O$.

```
100 INPUT I$
210 LET T$=O$: GO SUB 5000: LET
SP=IS: IF SP>0 THEN  PRINT O$;"
";
```

Cada objeto (complemento directo) puede compararse de la misma manera formando un bucle pero es esencial el restablecimiento del marcador de longitud de palabras —WL— a la correspondiente que se analiza en cada momento. (Téngase en cuenta que poniendo un punto y coma después de O$ se asegura que cada palabra se sitúe en la misma línea.)

```
200 RESTORE : FOR M=1 TO 6
205 READ O$: LET WL=LEN (O$)
220 NEXT M
```

Pueden hacerse pruebas similares para la identificación de palabras de la lista de adverbios y adjetivos. Obsérvese que RESTORE (seguido de número de línea) se utiliza para que no sea preciso leer, mediante la instrucción READ, los datos en el orden en que se entraron.

```
300 RESTORE 9060: FOR M=1 TO 3
305 READ V$: LET WL=LEN (V$)
310 LET T$=V$: GO SUB 5000: LET
```

```
SP=IS: IF SP>0 THEN  PRINT V$;"
";
320 NEXT M
400 RESTORE 9030: FOR M=0 TO 5
405 READ J$: LET WL=LEN (J$)
410 LET T$=J$: GO SUB 5000: LET
SP=IS: IF SP>0 THEN  PRINT J$;"
"
420 NEXT M
500 LET I$=""
510 PRINT "OTRA ENTRADA?"
520 INPUT Q$
530 CLS
540 GO TO 100
```

Para informar que se han hallado las palabras que utilizaremos posteriormente, se almacenarán en una matriz en el momento en que se encuentran. Ya tenemos una matriz de cadena —W$— para guardar las palabras y la ampliaremos hasta una capacidad de 20 (lo que tendría que ser suficiente para una frase un poco complicada).

```
10 DIM W$(20,12)
20 LET WC=1
```

Si se encuentra identidad o coincidencia, una variable de cadena (E$) con carácter temporal se pone al valor del orden de la palabra encontrada y se llama a la subrutina de la línea 1500 que la pone en el primer elemento de la citada matriz (véase el Diagrama de flujo 3.7).

```
 210 LET T$=O$:GO SUB 5000:LET S
P=IS:IF SP>0THEN LET E$=O$:PRINT
 E$;" ";:GO SUB 1500
1500 LET W$(WC)=E$
```

El contador de palabras WC se incrementa entonces para que la siguiente se ponga en el elemento que sigue antes de hacer el retorno de la subrutina.

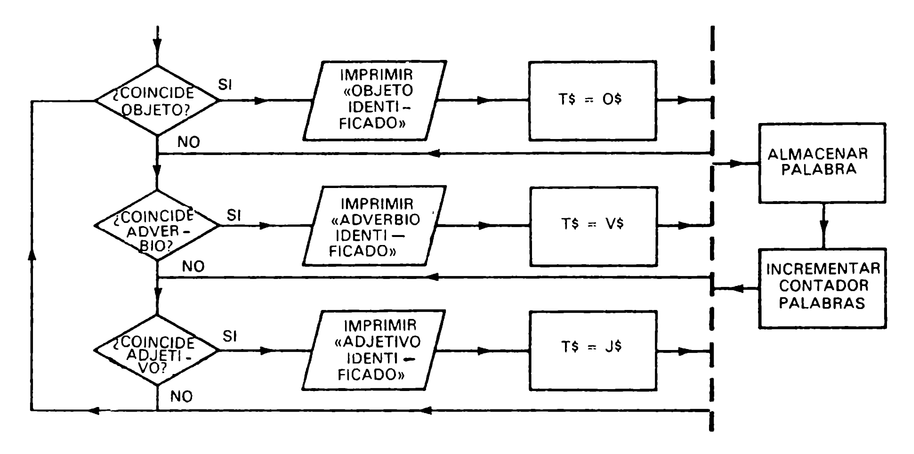


Diagrama de flujo 3.7. Análisis corrido

```
1520 LET WC=WC+1
1530 RETURN
```

Usando la cadena temporal E$ en la rutina de almacenamiento podremos también utilizarla en la prueba de adverbios y adjetivos exactamente igual.

```
 310 LET T$=V$: GO SUB 5000: LET
 SP=IS: IF SP>0 THEN LET E$=V$:
PRINT E$;" ";: GO SUB 1500
 410 LET T$=J$: GO SUB 5000: LET
 SP=IS: IF SP>0 THEN  LET E$=J$:
 PRINT E$;" ";: GO SUB 1500
```

La comparación parcial

Una ventaja del análisis corrido es que se puede reconocer una serie de palabras relacionadas sólo investigando los caracteres esenciales. Esto es obviamente útil, ya que evita tener que poner los nombres en singular y en plural como BOLLO y BOLLOS. Si se enmienda la línea 9010 de datos, como se muestra a continuación, se logrará la coincidencia o identidad para ambos casos.

```
9010 DATA "BIZCOCHO","BOLLO","PA
STEL"
9020 DATA "CAFE","TILA","PAN"
```

Sin embargo, las cosas no son tan sencillas. Poniendo BOLL se detectará BOLLO y BOLLOS, pero también podemos sacar BOLLA, BOLLADURA, BOLLON, BOLLICIO, etc.

Este problema no sólo se limita a los prefijos ya que el ordenador tampoco distinguirá entre PERO y APERO, por ejemplo. Se puede incluir una comprobación de que el carácter delante de cada identidad sea un espacio (es decir, que tal identidad sea una palabra, véase el Diagrama de flujo 3.8). La variable SP da la posición del principio de la palabra por lo que I$(SP — 1) es el carácter anterior a ella.

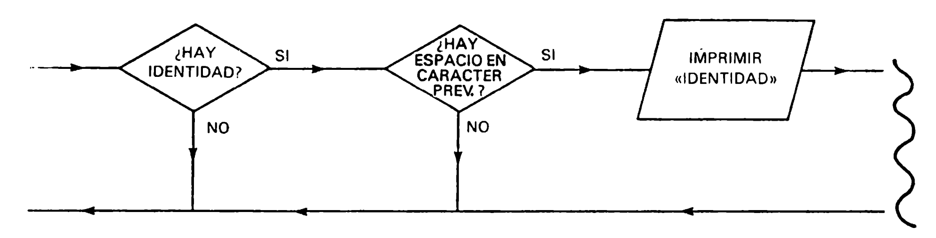


Diagrama de flujo 3.8. Comprobación de que se trata del principio de una palabra

```
210 LET T$=O$: GO SUB 5000: LET
SP=IS: IF SP=0 THEN NEXT M: GO
TO 300
211 IF I$(SP-1)<>" " THEN NEXT
M: GO TO 300
212 LET E$=O$: PRINT E$;" ";: G
O SUB 1500
310 LET T$=V$: GO SUB 5000: LET
SP=IS: IF SP=0 THEN NEXT M: GO
TO 400
311 IF I$(SP-1)<>" " THEN NEXT
M: GO TO 400
312 LET E$=V$: PRINT E$;" ";: G
O SUB 1500
```

```
 410 LET T$=J$: GO SUB 5000: LET
 SP=IS: IF SP=0 THEN  NEXT M: GO
 TO 500
 411 IF I$(SP-1)<>" " THEN  NEXT
 M: GO TO 500
 412 LET E$=J$: PRINT E$;" ";: G
O SUB 1500
```

Para que esto funcione correctamente con la primera palabra añádase un espacio al principio de I$.

```
110 LET I$=" "+I$
```

En forma similar se podrían hacer pruebas sobre la letra siguiente después de establecer la identidad de letras, o sobre la longitud de la palabra, para limitar las ya conocidas.

Una rutina de clasificación para efectuar reordenaciones

Aunque ya hemos detectado todas las palabras de la frase con independencia de su posición o cualquier otra cosa, se han encontrado y almacenado en el orden en que aparecían en la instrucción DATA. Esto es así porque la comparación empieza con el primer elemento en la lista de objetos en vez de empezar con la primera palabra de la frase. Sería útil si pudiéramos reordenar la matriz de palabras para que éstas estuvieran en el orden en que aparecen en la frase. Para lograrlo tenemos que registrar la posición de la palabra en la frase con la variable SP y los valores del contador de palabras WC cuando cada una se identifique en una nueva matriz de posición P(n,n). Se trata de una matriz bidimensional con la posición de la frase en el primer elemento, P(WC,0), y el Contador de palabras, P(WC,1), en el segundo.

```
  15 DIM P(20,2)
1510 LET P(WC,1)=SP: LET P(WC,2)
=WC
```

La rutina de clasificación que efectúa la reordenación se encuentra en la línea 4000 y sólo se va a ella cuando se logra una identificación.

```
 440 IF WC=0 THEN  GO TO 470
 450 GO SUB 4000
 460 GO TO 500
 470 PRINT "NO SE HA ENCONTRADO
IDENTIDAD"
```

La rutina de clasificación (véase el Diagrama de flujo 3.9) toma la posición en la frase de la primera palabra encontrada (primer elemento en la primera dimensión P(1,1)) y lo compara con la posición de la segunda palabra hallada (segundo elemento de la primera dimensión P(1 + 1,1)). Si la variable de posición para la primera palabra no tiene un valor más alto que el de la segunda, entonces esta se encuentra antes que aquella y debe invertirse el orden. Con ello se ponen los punteros de posición de frase en orden correcto pero los marcadores del contador de palabras también necesitan ser reordenados. El proceso se repite hasta que los punteros de palabras estén en orden correcto. Obsérvese que los contenidos reales de la matriz que alberga las palabras no se alteran. Unicamente se alteran sus punteros (los índices).

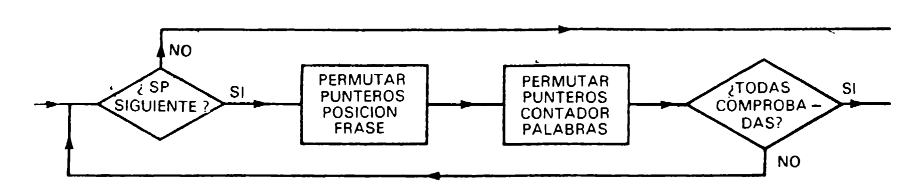


Diagrama de flujo 3.9. Clasificación de las palabras

```
4000 FOR N=1 TO WC-2
4010 IF P(N,1)<P(N+1,1) THEN  NE
XT N: GO TO 4040
4020 LET D=P(N,1): LET P(N,1)=P(
N+1,1): LET P(N+1,1)=D
```

```
4030 LET D=P(N,2): LET P(N,2)=P(
N+1,2): LET P(N+1,2)=D: GO TO 40
00
```

Si las cadenas se presentan en pantalla ahora en el orden variado del contador de palabras (WC), se hallarán como en la frase original, lo que hará más fácil su comprensión.

```
4040 PRINT : FOR N=1 TO WC-1
4050 PRINT W$(P(N,2));" ";
4060 NEXT N: PRINT
```

Todos los elementos de la matriz de posición de frase P(N,1) y del contador de palabras, WC, deben restablecerse a cero antes de la siguiente entrada.

```
4070 FOR N=1 TO 20
4080 LET P(N,1)=0
4090 NEXT N
4100 LET WC=1
4110 RETURN
```

4

La programación de respuestas

Hemos considerado en profundidad cómo analizar frases que se aplican al ordenador pero los mensajes que se han producido hasta ahora han sido muy limitados y rígidos. Aunque la mayoría de las palabras originales de la frase se usan frecuentemente en las respuestas, en una conversación real se considera el sujeto de la oración y se modifica de acuerdo con el contexto de la respuesta.

Por ejemplo:

YO NECESITO DESCANSO

pudiera esperar una respuesta afirmatoria:

TU NECESITAS DESCANSO

y en forma similar:

TU NECESITAS DESCANSO

daría lugar a:

YO NECESITO DESCANSO

Si se examina esta situación lógicamente, se aprecia que para cada sujeto en la oración entrada hay un sujeto equivalente de salida. Prácticamente lo que hemos hecho, por lo que se refiere al sujeto,

es permutarlo y añadir el resto de la frase en concordancia con el nuevo sujeto (y se ha modificado el verbo).

El pronombre «YO» son dos letras que se pueden poner en la variable I$. Si I$(TO2) = «YO» se ordena que «TU» se agregue delante del resto de la frase, previa modificación adecuada de la forma del verbo que ha de pasar de la primera persona a la segunda como se expresa en el listado siguiente: (I$ = «YO NECESITO DESCANSO»).

```
   10 INPUT I$
   30 IF I$( TO 2)="YO" THEN  PRI
NT "TU"+I$(3 TO 10)+"AS"+I$(12 T
O )
   60 GO TO 10
```

De análoga forma se pueden investigar los dos primeros caracteres, I$(TO2), con respecto a la palabra «TU» y, si es el caso, sustituirse por el pronombre «YO».

```
   50 IF I$( TO 2)="TU" THEN  PRI
NT "YO"+I$(3 TO 10)+"O"+I$(13 TO
 )
```

Si se prueba esto con una serie de frases se verá su utilidad hasta que empleemos el verbo auxiliar SER o ESTAR. Por ejemplo:

TU ESTAS CANSADO

que nos obligaría a analizar el verbo SER relacionando las expresiones YO ESTOY y TU ESTAS. Supongamos que a la afirmación «YO ESTOY CANSADO» se desea la respuesta «TU ESTAS CANSADO». Obsérvese que al final de estas instrucciones se ha de poner la que nos retorna a la entrada para evitar el confusionismo que se pudiera originar si se analizaran los pronombres personales de las líneas siguientes:

```
   20 IF I$( TO 8)="YO ESTOY" THE
N  PRINT "TU ESTAS"+I$(9 TO ): G
O TO 10
   40 IF I$( TO 8)="TU ESTAS" THE
N  PRINT "YO ESTOY"+I$(9 TO ): G
O TO 10
```

Listado resumen

```
   10 INPUT I$
   20 IF I$( TO 8)="YO ESTOY" THE
N  PRINT "TU ESTAS"+I$(9 TO ): G
O TO 10
   30 IF I$( TO 2)="YO" THEN  PRI
NT "TU"+I$(3 TO 10)+"AS"+I$(12 T
O )
   40 IF I$( TO 8)="TU ESTAS" THE
N  PRINT "YO ESTOY"+I$(9 TO ): G
O TO 10
   50 IF I$( TO 2)="TU" THEN  PRI
NT "YO"+I$(3 TO 10)+"O"+I$(13 TO
 )
   60 GO TO 10
```

Aunque así logramos nuestro propósito, el programa pronto se hará excesivamente largo ya que se necesita una instrucción por cada posibilidad, dado que hemos de tener en cuenta la longitud de la palabra o frase a analizar. Cuando se necesita la comparación de muchas palabras es mejor leerlas mediante la instrucción READ, del contenido de las DATA y efectuar la mencionada comparación por medio de un bucle. Se evitan errores si las palabras entradas y las correspondientes de salida, o frases, se entran en las líneas DATA por parejas hermanadas y se leen, con una instrucción READ, por orden en la matriz correspondiente. Empecemos un nuevo programa con estas líneas que establecen los datos.

```
9000 DATA "YO","TU","TU","YO","Y
O ESTOY","TU ESTAS","TU ESTAS","
```

```
YO ESTOY","YO SOY","TU ERES","TU
 ERES","YO SOY"
```

Volveremos a utilizar una búsqueda de cadenas que emplee la técnica de cadena deslizante dentro de un bucle que, de momento sólo presentará en pantalla la correspondiente palabra o frase, O$, que se adapta a la expresión entrada, T$ (véase el Diagrama de flujo 4.1). Una ventaja de este tipo de investigación es que encajará fácilmente los espacios de las frases ya que no se descompone la entrada I$ en palabras antes de la adaptación.

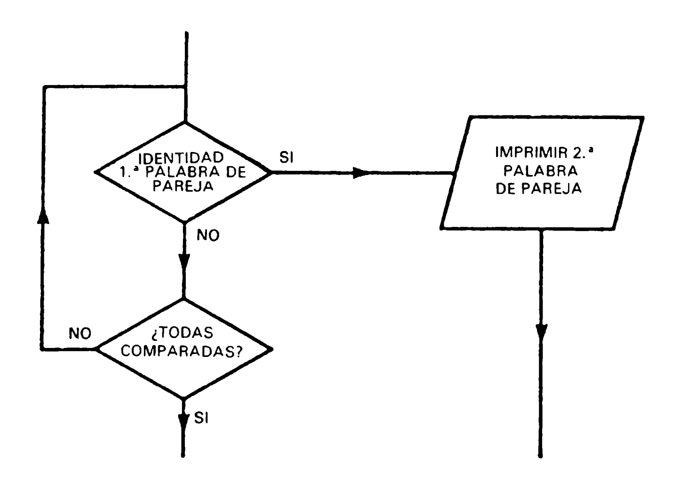


Diagrama de flujo 4.1. Utilización de la respuesta coincidente

```
 100 INPUT I$: LET I$=I$+" "
 110 LET ST=1
 200 RESTORE : FOR M=1 TO 4
 205 READ N$,O$: LET WL=LEN (N$)
 210 LET T$=N$: GO SUB 5000: LET
 SP=IS: IF SP>0 THEN  PRINT O$:
GO TO 100
 220 NEXT M
 250 GO TO 100
```

Si ahora se aplica cualquier frase que contenga la palabra «TU», por ejemplo, el ordenador responderá con la que tiene asociada («YO»).

Resulta mejor volver a definir la palabra respuesta requerida en una nueva cadena que constituye la primera parte de la respuesta —R$—, y después imprimirla haciendo PRINT cuando se sale del bucle.

```
210 LET T$=N$: GO SUB 5000: LET
SP=IS: IF SP>0 THEN  LET R$=O$:
GO TO 230
230 PRINT R$
```

Para tener una respuesta más completa se puede poner el resto de la frase, que permanece invariable, en otra cadena —S$— después de insertar un espacio (véase el Diagrama de flujo 4.2). No es difícil definir el «resto de la frase». Se necesita restar la posición final de la palabra de la longitud de la expresión. Recuérdese que la variable SP señala el principio de la palabra asociada cuya longitud (LEN) se halla registrada en la variable WL. Por consiguiente, el final de la frase es: I$(SP + WL TO).

```
210 LET T$=N$: GO SUB 5000: LET
SP=IS: IF SP=0 THEN  GO TO 220
215 LET R$=O$: LET S$=" "+I$(SP
+WL TO )
230 PRINT R$;S$
```

Ahora podemos probar:

YO SOY INTELIGENTE

y el ordenador está de acuerdo:

TU ERES INTELIGENTE

Pero no es muy cierto pues no hemos descargado las variables alfanuméricas I$, R$ y S$ antes de iniciar el bucle para la nueva entrada.

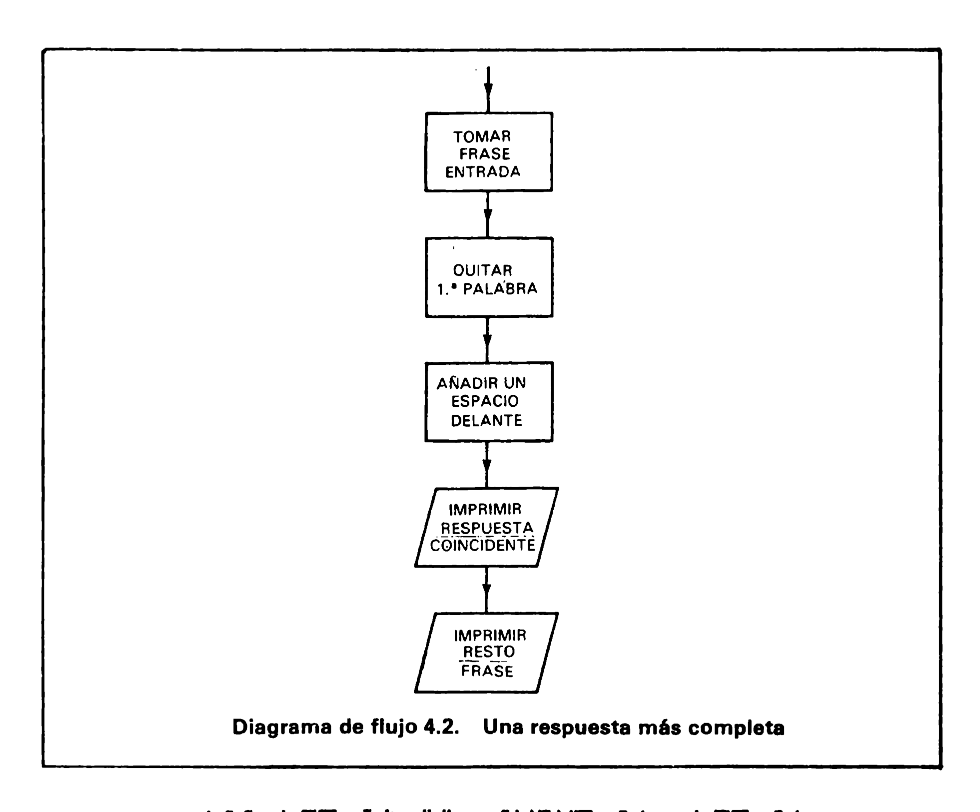


Diagrama de flujo 4.2. Una respuesta más completa

```
 100 LET I$="": INPUT I$: LET I$
=I$+" "
 240 LET R$="": LET S$=""
```

Antes de que nos creamos demasiado inteligentes, probemos:

SOMOS PURO ESPIRITU

que puede sorprendernos cuando nos responda:

YO

Si pensamos un poco en el problema nos daremos cuenta que nuestra palabra clave se encuentra dentro de otra palabra en esta frase en particular (la última sílaba de espíritu es el pronombre personal TU, de aquí que nos responda con YO).

Aunque cada palabra clave se va analizando por orden, sólo entra en R$ cuando se encuentra identidad o coincidencia, por lo que únicamente se informa la última que cumple esta condición. Dado que la palabra clave sólo se comprueba una vez en cada frase, las sílabas YO o TU originan problemas cuando no son la palabra esencial de la frase.

Para resolver esto hay que considerar cuáles son las palabras clave que pueden producir problemas. Debemos tratar todas las palabras clave de la misma manera, por lo que añadiremos un espacio a cada una. Obsérvese que no hay necesidad de añadir espacios al final de las respuestas.

```
9000 DATA "YO ","TU","TU ","YO",
"YO ESTOY ","TU ESTAS","TU ESTAS
 ","YO ESTOY","YO SOY ","TU ERES
","TU ERES ","YO SOY"
```

También necesitamos sustraer un carácter menos de I$ para lograr S$ ya que el espacio ha entrado a formar parte de la palabra clave.

```
 215 LET R$=O$: LET S$=" "+I$(SP
+WL TO )
```

Ahora el ordenador estará de acuerdo sobre la pureza de espíritu.

Si la primera palabra clave no se encuentra al principio de la frase, todo lo que hay delante de ella será ignorado en la respuesta. Por ejemplo, la réplica a:

¿QUE PASA SI YO ESTOY HABLANDO?

será:

¿TU ESTAS HABLANDO?

Sin embargo, añadiendo más palabras clave adecuadas en instrucciones DATA se lograrán mejores resultados, aunque algunas combinaciones no sean aceptables.

```
 200 RESTORE : FOR M=1 TO 6
9010 DATA "NOSOTROS ","NOSOTROS"
,"ELLOS ","ELLOS"
```

La adopción de decisiones lógicas por parte del ordenador

Hasta ahora nuestro ordenador sólo ha demostrado una inteligencia ligeramente superior a la de un loro ya que sólo ha devuelto una versión muy parecida a la que se ha entrado. La fase siguiente, en consecuencia, será la adopción de algunas decisiones lógicas en las respuestas derivadas de la expresión que se aplique.

Se definen variables para acoger el número de sujetos —SU—, verbos —VB— y respuestas —RP—, de manera que puedan desarrollarse fácilmente los programas. Con SU se define el número de sujetos reconocidos en las frases de entrada y salida, con VB los verbos y RP contiene las correspondientes respuestas.

```
  10 LET SU=14: LET VB=7: LET RP
=7
```

Las primeras dos líneas de instrucciones DATA contienen sujetos de entrada y salida emparejados (véase la Tabla 4.1). Como los pronombres personales (Yo, Tu, etc.), están frecuentemente encadenados a otras palabras para formar frases (como YO HE o YO TENGO), estas formas combinadas también se incluyen como datos. Obsérvese que están ordenados de tal forma que se hallará en primer lugar la frase más completa que contenga una palabra clave. Se añade un espacio al final de cada elemento para evitar problemas de adaptaciones parciales y se forma automáticamente un espacio en la respuesta.

```
9000 DATA "YO HE ","TU HAS ","TU
 HAS ","YO HE ","TU ","YO ","YO
","TU ","NOSOTROS HEMOS ","VOSOT
ROS HABEIS ","VOSOTROS HABEIS ",
"NOSOTROS HEMOS ","VOSOTROS ","N
OSOTROS ","NOSOTROS ","VOSOTROS
","EL HA ","EL HA ","ELLA HA ","
ELLA HA ","EL ","EL ","ELLA ","E
LLA ","ELLOS HAN ","ELLOS HAN ",
"ELLOS ","ELLOS "
```

Tabla 4.1. Pares de sujetos de la variable SU(N,N)

ENTRADA	SALIDA
YO HE	TU HAS
YO TENGO	TU TIENES
YO SOY	TU ERES
YO ESTOY	TU ESTAS
TU HAS	YO HE
TU TIENES	YO TENGO
TU ERES	YO SOY
TU ESTAS	YO ESTOY
TU	YO
ELLA TIENE	ELLA TIENE
ELLA ES	ELLA ES
ELLA ESTA	ELLA ESTA
ELLA HA	ELLA HA
ELLOS HAN	ELLOS HAN
ELLOS SON	ELLOS SON
ELLOS ESTAN	ELLOS ESTAN
ELLOS	ELLOS
EL HA	EL HA
EL TIENE	EL TIENE
EL ES	EL ES
EL ESTA	EL ESTA
EL	EL
VOSOTROS HABEIS	NOSOTROS HEMOS
VOSOTROS TENEIS	NOSOTROS TENEMOS
VOSOTROS SOIS	NOSOTROS SOMOS
VOSOTROS ESTAIS	NOSOTROS ESTAMOS
ELLA	ELLA
NOSOTROS TENEMOS	VOSOTROS TENEIS
NOSOTROS HEMOS	VOSOTROS HABEIS
NOSOTROS SOMOS	VOSOTROS SOIS
NOSOTROS ESTAMOS	VOSOTROS ESTAIS
VOSOTROS	NOSOTROS
NOSOTROS	VOSOTROS
YO	TU

Nota: No se han puesto todos en el listado para no hacer el programa demasiado largo.

La siguiente línea DATA contiene los verbos principales. Los verbos auxiliares, ser, estar, haber y tener, se han omitido premeditadamente ya que muchas de sus formas se incluyen con los sujetos. Ampliando datos se podrían completar en todas sus formas y conjugaciones.

```
9060 DATA "ODIO ","AMO ","TRABAJ
ADO ","SIENTO ","PAREZCO ","TENI
DO ","SE "
```

El último conjunto de datos contiene las respuestas. Para hacer las cosas fáciles de entender y adaptar en esta fase todas las respuestas contienen el verbo original, aunque, por supuesto, puede decirse cualquier cosa.

```
9070 DATA "TAMBIEN,Y EN ESPECIAL
 A LOS ENGREIDOS ","MUCHO MAS ",
"HECHO EL VAGO ","LO MISMO ","TA
MBIEN COMO MUCHOS ","TENIDO CONS
TANCIA ","TAMPOCO,DADAS LAS CIRC
UNSTANCIAS "
```

Cómo se halla la coincidencia o identidad entre los elementos comparados

La primera variable SU empareja los datos que son leídos y la cadena de entrada se compara con la lista de sujetos —N$— (Véase el Diagrama de flujo 4.3). Si no existe identidad o coincidencia en la comparación, si no se corresponden cadena y sujeto, se pide otra entrada o bien la variable alfanumérica A$ se hace igual a la palabra idéntica mientras B$ toma la recíproca. Una variable de coincidencia o identidad de sujeto —SM— adquiere el valor del número de orden del sujeto con el que se corresponde —M—.

```
  10 LET SU=14: LET VB=7: LET RP
=7
 100 LET I$=" ": INPUT I$: PRINT
```

```
I$: LET I$=I$+" "
 105 IF I$( TO 4)="ALTO" THEN  P
RINT "ALTO": STOP
 200 RESTORE 9000: FOR M=1 TO SU
 210 READ N$,O$: LET WL=LEN (N$)
 220 LET T$=N$: GO SUB 5000: LET
 SP=IS: IF SP>0 THEN  LET A$=N$:
 LET B$=O$: LET SM=M: GO TO 300
 230 NEXT M
 240 GO TO 100
```

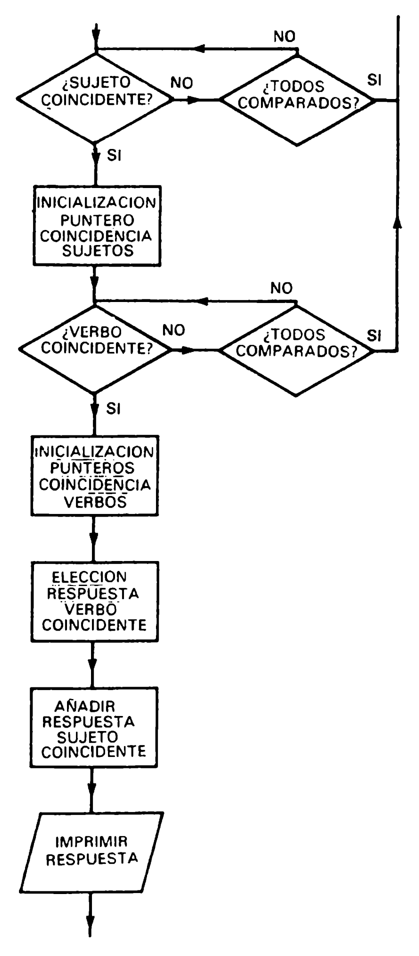


Diagrama de flujo 4.3. Establecimiento de los punteros de coincidencia

Todos los verbos (VB) de las instrucciones DATA se comparan con I$. Si no se encuentra el verbo, se rechaza la entrada y si se da con él se hace igual a M la variable de identidad o coincidencia VM. Obsérvese que el puntero de datos (DATA) debe restablecerse (RESTORE) en la línea 9060 cuando se salta de la lista de sujetos. (Los verbos se ponen en la forma en que aparecerán en las frases de entrada para permitir la identidad.)

```
 300 RESTORE 9060: FOR M=1 TO VB
 310 READ V$: LET WL=LEN (V$)
 320 LET T$=V$: GO SUB 5000: LET
 SP=IS: IF SP>0 THEN  LET VM=M:
GO TO 500
 330 NEXT M
```

La confección de las respuestas

Ahora que se ha hallado la coincidencia o identidad entre el sujeto y el verbo, se puede escoger la respuesta apropiada utilizando VM como puntero en los datos de respuesta que empiezan en la línea 9070.

```
 500 RESTORE 9070: FOR N=1 TO VM
: READ R$: NEXT N
```

En el caso más sencillo basta con añadir el sujeto apropiado (A$) delante de R$ antes de ordenar su presentación.

```
520 LET R$=A$+R$
550 PRINT R$
560 GO TO 100
```

Ahora, por ejemplo, si se introduce la frase:

YO ODIO LOS ORDENADORES

el programa responderá:

YO TAMBIEN, Y EN ESPECIAL A LOS ENGREIDOS

y:

YO PAREZCO MUY LENTO

la respuesta generada será:

YO TAMBIEN, COMO MUCHOS

El cambio de la variable del sujeto para simplificar el procedimiento

Si se prefiere que la máquina esté de acuerdo con nosotros en lugar de tratar de apabullarnos, basta con cambiar la variable del sujeto —A$— por la que contiene el segundo elemento de la pareja (B$).

```
520 LET R$=B$+R$
```

Si ponemos ahora:

NOSOTROS HEMOS TRABAJADO MUCHO

Se responde:

VOSOTROS HABEIS HECHO EL VAGO.

Hay que tener en cuenta que el verbo en la lista correspondiente ha de estar en la misma forma que en la pregunta planteada ya que la coincidencia se establece entre ambos y sólo se logrará cuando los dos sean iguales. La respuesta, sin embargo, se puede poner como se desee. Si la frase es:

YO SE MUCHO

la lista de verbos ha de incluir: ...«SE»...; y la variable que contiene

las respuestas —R$— llevará la réplica que corresponde al mismo orden en que está la citada forma verbal. La respuesta pudiera ser:

TU TAMPOCO, DADAS LAS CIRCUNSTANCIAS

Para una forma con más variedad podríamos escoger al azar uno de los dos sujetos emparejados con las siguientes líneas del listado pero esto nos acarrearía una nueva explicación ya que el verbo cambia al variar la persona. Sólo sería aceptable cuando la respuesta fuera poco precisa, como sucede en los ejemplos.

```
510 LET RS=INT (2*RND)
515 IF RS=0 THEN  LET R$=A$+R$:
GO TO 550
530 LET R$=B$+R$
```

Sería posible procesar las respuestas cortando su cadena por el verbo, separando la raíz y la terminación para incorporar la que estuviera acorde con el sujeto, pero ello alargaría considerablemente el programa en esta fase.

La adaptación del sujeto al contexto

Si la elección del sujeto se hiciera de acuerdo con el contexto de la respuesta, el procedimiento sería más adecuado, pero para hacerlo necesitamos unos marcadores para ella. Vamos a utilizar la barra (/) para indicar que seleccionamos el primer elemento de la pareja de sujetos y un asterisco (*) para el segundo.

```
9070 DATA "/CREO QUE TE ODIO A T
I TAMBIEN ","/LAS AMO MAS QUE TU
 ","*ESTUDIADO MUCHO ","*NO SIEN
TES UN PIMIENTO ","*PARECES UNA
TORTUGA ","*HECHO EL VAGO ","*NO
 SABES NADA "
```

Se puede buscar en la cadena R$ señalada por el marcador de orden de los verbos —VM— la existencia de la barra (/), siempre que la pongamos bajo otro nombre como I$, antes de aplicar la rutina de coincidencia (INSTR). Si se encuentra la barra, se añade el contenido del primer elemento del par de sujetos (A$) a la respuesta R$, sustrayendo el primer carácter (la barra). (Véase el Diagrama de flujo 4.4.)

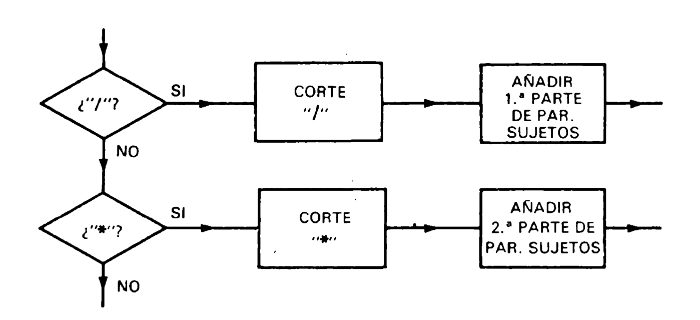


Diagrama de flujo 4.4. Adecuación del sujeto al contexto

```
 510 LET I$=R$: LET WL=1: LET T$
="/": GO SUB 5000
 520 IF IS>0 THEN  GO TO 800
 800 LET R$=A$+R$(2 TO )
 810 GO TO 530
```

Si no se encuentra la barra en la respuesta se hace una nueva búsqueda para encontrar el asterisco y, si se da con él, se utiliza el segundo elemento de la pareja de sujetos (B$) de análoga forma.

```
530 LET T$="*": GO SUB 5000
540 IF IS>0 THEN  GO TO 820
820 LET R$=B$+R$(2 TO )
830 GO TO 550
```

El ejemplo:

YO AMO LAS PLANTAS

nos dará

YO LAS AMO MAS QUE TU

y

YO HE TRABAJADO INTENSAMENTE

producirá:

TU HAS ESTUDIADO MUCHO

La frase:

YO ODIO A LOS ORDENADORES

dará origen a la réplica:

YO CREO QUE TE ODIO A TI TAMBIEN

La inserción del sujeto en las frases

Para hacer las cosas más sencillas siempre hemos empezado las respuestas con el sujeto pero en la vida real normalmente no es este el caso. Ahora que ya tenemos los marcadores en las respuestas para indicar el tipo de sujeto que se ha de incorporar, también podemos usarlos para señalar donde insertarlo. Primero, modificaremos la instrucción DATA para que la palabra que se va a introducir no esté nunca en primer lugar ya que entonces no ha lugar a la inserción.

```
9070 DATA "ESTO ME HACE PENSAR Q
UE CREO QUE / TE ODIO A TI TAMBI
EN ","PERO / LAS AMO MAS QUE TU
","PERO ES POSIBLE QUE * ESTUDIA
DO MUCHO ","NO PRESUMAS DE OIDO
FINO QUE * NO SIENTES NI EL ESTA
MPIDO DE UNA TRACA ","PERDONA,PE
```

```
RO * PARECES UNA TORTUGA ","YO C
REO MAS BIEN QUE * HECHO EL VAGO
 ","NO PRESUMAS QUE * NO SABES N
ADA "
```

Ya tenemos un registro para hacer la inserción puesto que la variable IS nos dice donde se encuentra dentro de la respuesta la barra o el asterisco. Todo lo que precisamos es separar la parte de la respuesta que está delante del marcador, R$(TO IS — 1), añadir la versión adecuada del sujeto (A$ o B$) y poner a continuación la parte restante, R$(IS + 1 TO).

```
 800 LET R$=R$( TO IS-1)+A$+R$(I
S+1 TO )
 820 LET R$=R$( TO IS-1)+B$+R$(I
S+1 TO )
```

Ahora:

YO ODIO LOS ORDENADORES

produce:

ESTO ME HACE PENSAR QUE CREO QUE YO TE ODIO A TI TAMBIEN

y:

YO AMO LAS FLORES

nos da:

PERO YO LAS AMO MAS QUE TU

Aunque hemos puesto los sujetos en la respuesta de una forma más natural, sólo utilizamos uno en cada frase. Otra pequeña modificación nos permitirá poner cualquier número de ellos. Todo lo que

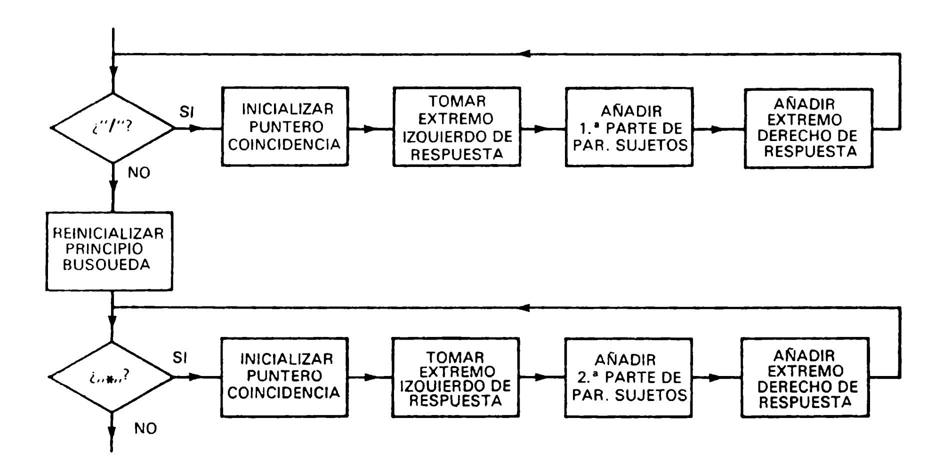


Diagrama de flujo 4.5. Inserción en una frase

hemos de hacer es mantener la búsqueda de los marcadores hasta que no se encuentre ninguno más. Una variable de iniciación, ST, recibe el valor 1 en la línea 505 y después se hace la búsqueda del primer tipo de marcador. Cuando se encuentra, se suma una unidad a la posición de coincidencia. Cuando R$ ha sido modificado por la línea 800, necesitamos regresar a la 510 para buscar más marcadores. Si no se encuentran más para el primer marcador se restablece a 1 nuevamente la variable ST. El segundo tipo de marcador se investiga de la misma manera.

```
 505 LET ST=1
 520 IF IS>0 THEN  LET ST=IS+1:
GO TO 800
 525 LET ST=1
 530 LET I$=R$: LET T$="*": GO S
UB 5000
 540 IF IS>0 THEN  LET ST=IS+1:
GO TO 820
 810 GO TO 510
 830 GO TO 530
9070 DATA "ESTO ME HACE PENSAR Q
UE CREO QUE / ODIO MUCHAS COSAS,
ESPECIALMENTE * ","PERO / LAS AM
```

```
O MAS QUE * ","PERO ES POSIBLE Q
UE * ESTUDIADO MUCHO,*SIDO CONST
ANTES ","NO PRESUMAS DE OIDO FIN
O QUE * NO SIENTES NI EL RUIDO D
E UNA TRACA;NI * NI / OIMOS BIEN
 ","PERDONA,PERO * PARECES UNA T
ORTUGA COMO /","YO CREO MAS BIEN
 QUE *HECHO EL VAGO ","NO PRESUM
AS,QUE *SABES COMO /:NADA "
```

Y ahora:

YO SIENTO EL RUMOR DEL AGUA

nos da:

NO PRESUMAS DE OIDO FINO QUE TU NO SIENTES NI EL RUIDO DE UNA TRACA, NI TU NI YO OIMOS BIEN

y:

NOSOTROS HEMOS TRABAJADO CON SUERTE

lo que nos da:

PERO ES POSIBLE QUE VOSOTROS HABEIS ESTUDIADO MUCHO, VOSOTROS HABEIS SIDO CONSTANTES

El caso de la divergencia entre el sujeto y el objeto

Parece que todo va marchando bien hasta que probemos algo como esto:

YO ODIO TU MIRADA

que nos da:

ESTO ME HACE PENSAR QUE TU ODIAS MUCHAS COSAS, ESPECIALMENTE YO

El problema surge porque nos salimos de la rutina de búsqueda tan pronto como se encuentra la primera identidad o coincidencia. Aunque andamos buscando el sujeto YO, encontramos primero el objeto TU ya que éste se halla antes que el pronombre personal YO en la cadena de sujetos y por consiguiente es hallado primero aunque ocupe el segundo lugar en la frase.

Como no podemos reproducir todos los recursos del cerebro humano, habrá que suponer que el sujeto siempre está delante del verbo y que el complemento objeto le sigue. En el programa hasta ahora hemos buscado el sujeto antes que el verbo por lo que tendremos que invertir el orden sustituyendo las líneas 200 a 240 por sus equivalentes, de la 400 a la 440. La posición del verbo en la entrada es el valor de SP cuando ya ha sido encontrado. Conservaremos dicho valor en una variable indicadora de posición del verbo (VP).

```
 320 LET T$=V$: GO SUB 5000: LET
 SP=IS: IF SP>0 THEN  LET VM=M:
LET VP=SP: GO TO 400
```

Ahora cuando se encuentra la identidad o coincidencia en la cadena de sujetos, comparamos su posición SP con VP y rechazamos la identidad si se halla después del verbo. (Véase el Diagrama de flujo 4.6.)

```
 400 RESTORE 9000: FOR M=1 TO SU
 405 READ N$,O$: LET WL=LEN (N$)
 410 LET T$=N$: GO SUB 5000: LET
 SP=IS: IF SP>0 AND SP<VP THEN
LET A$=N$: LET B$=O$: LET SM=M:
GO TO 500
```

Soluciones para la complejidad de diferentes formas verbales

Las múltiples formas del verbo en tiempos y personas crean un problema de gran complejidad que puede resolverse ampliando extraordinariamente el programa. Si no hay problema de memoria y de tiempo (si el programa se escribe en BASIC), es posible extender los

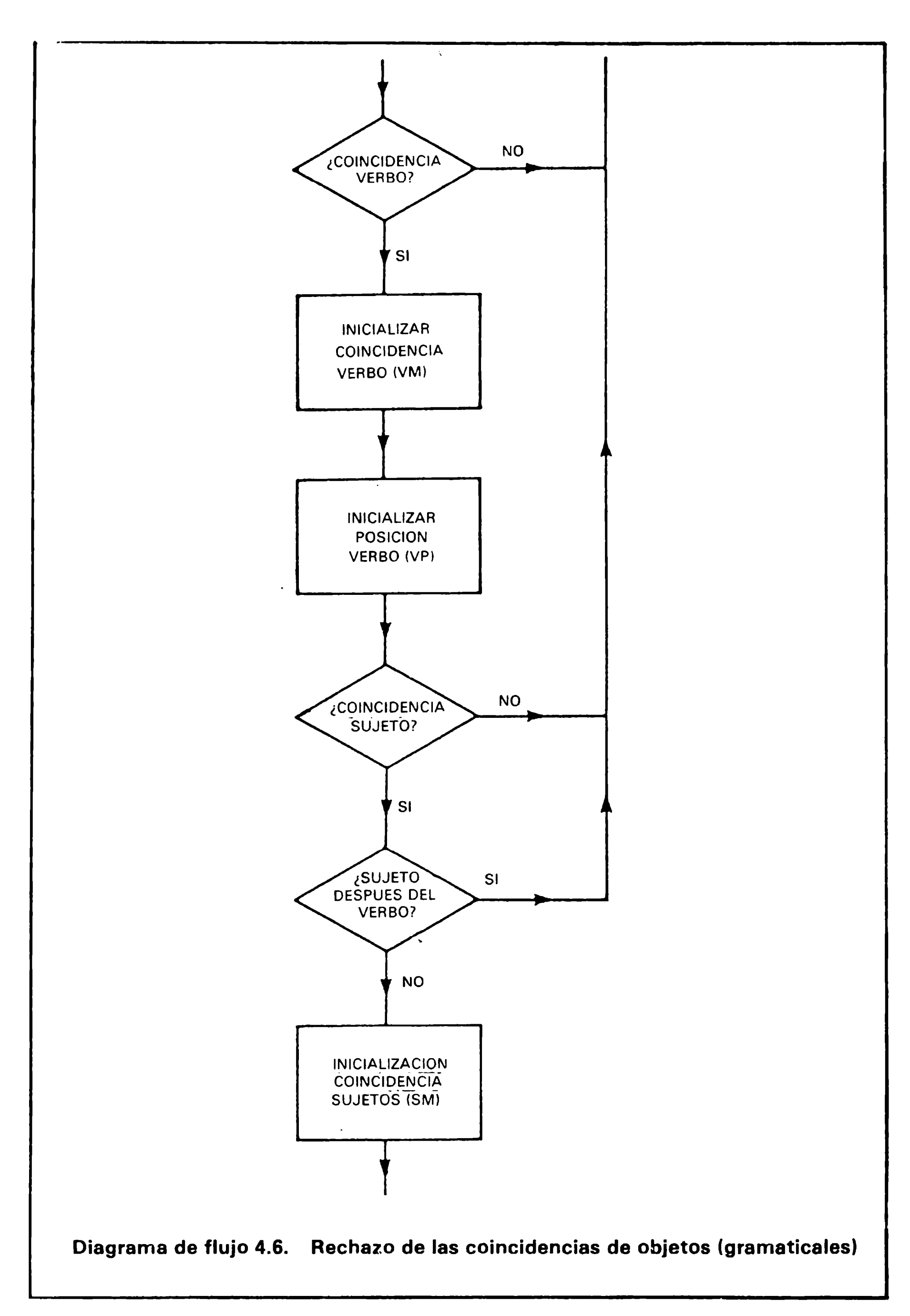


Diagrama de flujo 4.6. Rechazo de las coincidencias de objetos (gramaticales)

tiempos de los verbos hasta incluirlos a todos y aumentar igualmente las instrucciones DATA que contengan las respuestas para adaptarlas a aquellos. Como se ve, hay posibilidades pero se escapan a las de un equipo como el considerado en este libro que, por otra parte, sirve perfectamente para la presentación y estudio elemental de la cuestión.

La introducción de los verbos con todas sus personas, como se ha hecho en el listado, obliga a modificar la variable que sirve para marcar verbos —VM—. Esto ha de hacerse para que la posición encontrada de la forma en cuestión pueda ser dirigida a la respuesta correspondiente, so pena que preparemos una de ellas para cada persona del verbo, cosa que es posible y se brinda como idea al lector para que extienda el programa. Esta modificación consiste en que, como se han incluido todas las personas del presente de indicativo (6), habrá que dividir por este valor el de la variable VM, y como necesitamos un número entero que nos lleve a la respuesta correspondiente, introduciremos en la línea 320, además de las instrucciones que incorpora, LET VM = INT (M/6) + 1:LET VP = IS.

```
  10 LET SU=14: LET VB=42: LET R
P=7
9060 DATA "ODIO ","ODIAS ","ODIA
 ","ODIAMOS ","ODIAIS ","ODIAN "
,"AMO ","AMAS ","AMA ","AMAMOS "
,"AMAIS ","AMAN ","TRABAJADO ","
TRABAJADO ","TRABAJADO ","TRABAJ
ADO ","TRABAJADO ","TRABAJADO ",
"SIENTO ","SIENTES ","SIENTE ","
SENTIMOS ","SENTIS ","SIENTEN ",
"PAREZCO ","PARECES ","PARECE ",
"PARECEMOS ","PARECEIS ","PARECE
N ","TENIDO ","TENIDO ","TENIDO
","TENIDO ","TENIDO ","TENIDO ",
"SE ","SABES ","SABE ","SABEMOS
","SABEIS ","SABEN "
```

Como cierre del capítulo se incluye el listado completo en su forma final. No es un programa perfecto pues precisaría de una mejor adecuación de las respuestas posibles, de acuerdo con la persona en que se halla el sujeto en la frase de entrada (aumentar el número de res-

puestas). No cabe duda que la inclusión de más verbos permitiría un diálogo más interesante y la extensión de las respuestas también sería conveniente. Lo que se ha expuesto en el capítulo puede servir de guía para un mejor trabajo aunque es indudable que la tarea pendiente es de gran envergadura dada la estructura del idioma castellano.

```
  10 LET SU=14: LET VB=42: LET R
P=7
 100 LET I$=" ": INPUT "PONGA SU
 FRASE ";I$: PRINT 'I$: LET I$=I
$+" "
 105 IF I$( TO 4)="ALTO" THEN  P
RINT "ALTO": STOP
 300 RESTORE 9060: FOR M=1 TO VB
 310 READ V$: LET WL=LEN (V$)
 320 LET T$=V$: GO SUB 5000: LET
 SP=IS: IF SP>0 THEN  LET VM=INT
 (M/6)+1: LET VP=SP: GO TO 335
 330 NEXT M
 335 IF M/6=INT (M/6) THEN  LET
VM=VM-1
 400 RESTORE 9000: FOR M=1 TO SU
 405 READ N$,O$: LET WL=LEN (N$)
 410 LET T$=N$: GO SUB 5000: LET
 SP=IS: IF SP>0 AND SP<VP THEN
LET A$=N$: LET B$=O$: LET SM=M:
GO TO 500
 420 NEXT M
 500 RESTORE 9070: FOR N=1 TO VM
: READ R$: NEXT N
 505 LET ST=1
 510 LET I$=R$: LET WL=1: LET T$
="/": GO SUB 5000
 520 IF IS>0 THEN  LET ST=IS+1:
GO TO 800
 525 LET ST=1
 530 LET I$=R$: LET T$="*": GO S
UB 5000
 540 IF IS>0 THEN  LET ST=IS+1:
GO TO 820
```

```
 550 PRINT R$
 560 GO TO 100
 800 LET R$=R$( TO IS-1)+A$+R$(I
S+1 TO )
 810 GO TO 510
 820 LET R$=R$( TO IS-1)+B$+R$(I
S+1 TO )
 830 GO TO 530
5000 FOR N=1 TO LEN (I$)-WL+1
5010 IF I$(N TO (N+WL-1))=T$ THE
N  LET IS=N: RETURN
5020 NEXT N
5030 LET IS=0
5040 RETURN
9000 DATA "YO HE ","TU HAS ","TU
 HAS ","YO HE ","TU ","YO ","YO
","TU ","NOSOTROS HEMOS ","VOSOT
ROS HABEIS ","VOSOTROS HABEIS ",
"NOSOTROS HEMOS ","VOSOTROS ","N
OSOTROS ","NOSOTROS ","VOSOTROS
","EL HA ","EL HA ","ELLA HA ","
ELLA HA ","EL ","EL ","ELLA ","E
LLA ","ELLOS HAN ","ELLOS HAN ",
"ELLOS ","ELLOS "
9060 DATA "ODIO ","ODIAS ","ODIA
 ","ODIAMOS ","ODIAIS ","ODIAN "
,"AMO ","AMAS ","AMA ","AMAMOS "
,"AMAIS ","AMAN ","TRABAJADO ","
TRABAJADO ","TRABAJADO ","TRABAJ
ADO ","TRABAJADO ","TRABAJADO ",
"SIENTO ","SIENTES ","SIENTE ","
SENTIMOS ","SENTIS ","SIENTEN ",
"PAREZCO ","PARECES ","PARECE ",
"PARECEMOS ","PARECEIS ","PARECE
N ","TENIDO ","TENIDO ","TENIDO
","TENIDO ","TENIDO ","TENIDO ",
"SE ","SABES ","SABE ","SABEMOS
","SABEIS ","SABEN "
9070 DATA "ESTO ME HACE PENSAR Q
UE /TAMBIEN,COMO * ","PERO /MAS
QUE * ","POSIBLEMENTE *TRABAJADO
```

```
MUCHO,*PERSEVERADO ","NI HABLAR
,NI *NI /SENTIMOS MUCHO ","PERDO
NAR,PERO *IGUAL QUE /","YO CREO
MAS BIEN QUE *HECHO EL VAGO ","N
O PRESUMIR,QUE *IGUAL QUE /:NADA
 DE NADA"
```

Reglas para la construcción de las frases tal como está preparado el programa:

1. El verbo ha de estar en alguna de las formas del vocabulario (es posible aumentarlo para tener más capacidad de respuesta).
2. Son aconsejables las expresiones en primera o segunda persona, tanto en singular como en plural, pues las que se hagan en tercera persona tendrán una respuesta en la misma, lo que a veces tiene muy poco sentido.
3. Para los tiempos compuestos del verbo sólo pueden hacerse frases con «trabajado» y «tenido», las únicas formas del vocabulario (línea 9060).
4. Hay que poner las frases con letras mayúsculas pues así están en los vocabularios.

Indudablemente con estas ideas se puede extender el programa para darle más capacidad. Esto queda a la discreción del lector.

La línea 105 se ha introducido para salir del programa poniendo como entrada (INPUT) la palabra ALTO. También se sale si se pulsa ENTER como entrada.

Ejemplos de frases que pueden servir para probar el programa:

Yo odio los ordenadores
Tu amas las flores
Yo he trabajado intensamente
Nosotros sentimos el rumor del agua
Vosotros pareceis algo torpes
Ellos han tenido poca suerte
Nosotros sabemos mucha informática
Etc., Etc.

5

Los sistemas expertos

Un experto es una persona que sabe mucho sobre un determinado tema y está capacitado para dar adecuado asesoramiento sobre él (la opinión del experto). Tal capacidad sólo se adquiere después de un largo entrenamiento y dilatada experiencia por lo que los expertos son desgraciadamente pocos y además raras veces se hallan a mano cuando los problemas necesitan su intervención.

Los científicos han creado programas que realizan las funciones de tales expertos con la ventaja de poder reproducirse fácilmente para constituir un número prácticamente infinito que no necesitan hacer un alto para tomar el café, ni tienen sueño, ni solicitan aumentos de sueldo, etc. Por supuesto que el ordenador ha de ser totalmente lógico y sólo sigue las instrucciones creadas por el programador. Es interesante señalar que los autores de ciencia-ficción han imaginado ciertos problemas cuando el súmmum de la experiencia (como HAL de 2001: Odisea en el Espacio o los robots positrónicos de Isaac Asimov) se enfrenta con hechos no previstos que no encajan con más de uno de los supuestos principales y que si no llegan a provocar el colapso del sistema sí que originan traumas «pseudo-nerviosos».

Antes de empezar a hacer un programa para sistemas expertos debemos preguntarnos cómo actúa un hombre experimentado.

Consideremos primeramente la situación más sencilla, en la que un experto ha de encontrar la solución de un problema conocido. En primer lugar, pide la información que después compara con la que posee y trata de comprobar si existe correspondencia entre ellas. Finalmente expresa si se ha logrado o no tal correspondencia o identidad.

Lo que necesitamos pues es simplemente un programa de base de datos que trate de adaptar la información de entrada con la que tiene almacenada (véase el Diagrama de flujo 5.1). Un sistema amistoso aceptaría un lenguaje natural (véanse los capítulos anteriores) pero, para no complicar las cosas, nos sujetaremos a un formato de entrada fijo. Para empezar tratemos de reconocer los sonidos que emiten ciertos animales. Estableceremos parejas de datos en las que la primera palabra —Q$— contiene los sonidos conocidos y las segundas A$— los nombres de los animales.

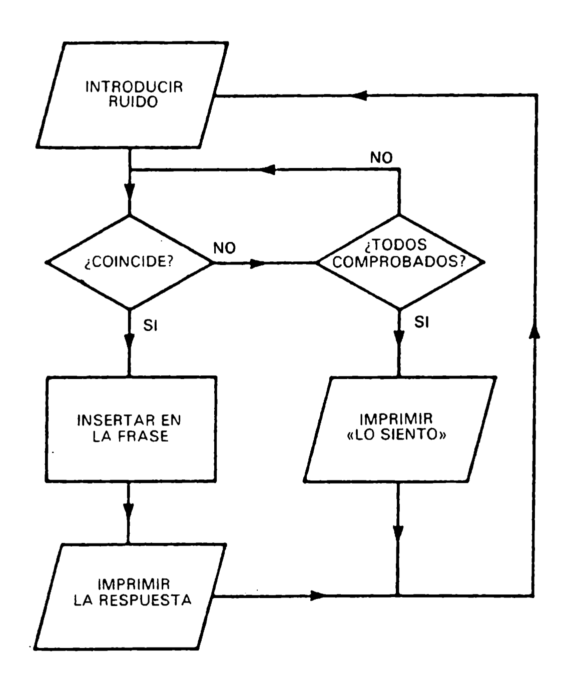


Diagrama de flujo 5.1. Un «experto» simplificado

```
9000>DATA " MAULLA","GATO"," LAD
RA","PERRO"," MUGE","BUEY"," ULU
LA","BUHO"," RELINCHA","CABALLO"
```

Ahora sólo necesitamos pedir un animal y compararlo con los contenidos en la variable Q$.

```
   5 REM RUTINA PARA ABRIR INTER
ROGANTE.(PULSAR LA "I" DE LA PRE
GUNTA EN MODO GRAFICO)
  10 RESTORE 10: FOR N=1 TO 7: R
EAD A: POKE USR "I"+N,A: NEXT N:
 DATA 2,0,2,14,16,17,14
  20 PRINT ''"IQUE SONIDO EMITE
EL ";
  30 INPUT I$: PRINT I$;"?"
  35 IF I$( TO 3)="FIN" THEN  ST
OP
  40 RESTORE 9000: FOR N=1 TO 5:
 READ Q$,A$: IF I$=A$ THEN  GO T
O 90
  50 NEXT N
  60 PRINT "LO SIENTO,NO SE QUE
RUIDO HACE  ESE ANIMAL"
  70 GO TO 20
  90 PAUSE 50
 100 PRINT "EL ";A$;Q$
 105 PAUSE 100
 110 GO TO 20
9000 DATA " MAULLA","GATO"," LAD
RA","PERRO"," MUGE","BUEY"," ULU
LA","BUHO"," RELINCHA","CABALLO"

IQUE SONIDO EMITE EL BUHO?
EL BUHO ULULA

IQUE SONIDO EMITE EL PERRO?
EL PERRO LADRA

IQUE SONIDO EMITE EL BUEY?
EL BUEY MUGE
```

Llegados a este punto quizás deberíamos decir que nuestro ordenador experto puede realizar su tarea mejor que un ser humano ya que no hace juicios subjetivos, no llega a aburrirse con su trabajo ni se olvida de buscar toda la información en su memoria.

Las comprobaciones secuenciales para hallar una solución

El ejemplo anterior es muy sencillo ya que sólo se hace una pregunta y únicamente hay una respuesta posible. En realidad, tenemos que enfrentarnos a problemas más difíciles en los que la respuesta no puede encontrarse sin hacer una serie de preguntas. Por ejemplo, ¿qué haría un experto si, al poner la llave del encendido de su coche y accionarla, no sucediera nada?

Podría haber varias causas que provocaran esta inactividad:

—BATERIA DESCARGADA
—MALAS CONEXIONES
—CONMUTADOR AVERIADO
—MOTOR DE ARRANQUE BLOQUEADO
—MOTOR DE ARRANQUE AVERIADO
—SOLENOIDE ROTO

Para encontrar la causa se seguiría un camino lógico y se harían unas comprobaciones. La primera cosa sería averiguar si sólo es el motor de arranque el que no funciona:

¿LUCE LA LAMPARA DEL ENCENDIDO? (S/N)

Si la respuesta es «N», no existe energía en el conmutador por lo que la causa puede ser una de las tres primeras posibilidades citadas anteriormente. Es posible acercarnos más averiguando si las luces del coche funcionan:

¿FUNCIONAN CORRECTAMENTE LAS LUCES? (S/N)

Si la respuesta es «S», la batería no puede estar descargada y su energía se conecta correctamente al conmutador de luces. En consecuencia lo que debe fallar es el conmutador de arranque. Ya se puede sugerir que se sustituya.

SUSTITUYA EL CONMUTADOR DE ENCENDIDO

Si las luces no funcionan, habrá que comprobar las conexiones.

¿ESTAN LAS CONEXIONES DE LA BATERIA EN BUEN ORDEN? (S/N)

Si se responde con un sí, la batería estará descargada y ya se puede decir que se cambie (o que se empuje el coche).

CARGUE LA BATERIA O EMPUJE EL COCHE

De análoga forma se puede hacer una secuencia de comprobaciones para resolver un caso en el que hay energía pero no funciona el mecanismo de arranque (las tres últimas posibilidades).

La forma más sencilla para programar esta estructura encadenada es por medio de una serie de instrucciones condicionales IF-THEN (véase el Diagrama de flujo 5.2).

```
   5 RESTORE : FOR N=1 TO 7: REA
D A: POKE USR "I"+N,A: NEXT N: D
ATA 2,0,2,14,16,17,14
  10 PRINT "DIAGNOSTICO DE LA AV
ERIA"
  20 PRINT
  30 PRINT ''"ILUCE LA LAMPARA D
EL ENCENDIDO?  (S/N)"
  40 INPUT I$
  50 IF I$="S" THEN  GO TO 180
  60 PRINT '"IFUNCIONAN CORRECTA
MENTE LAS"'"LUCES? (S/N)"
  70 INPUT I$
  80 IF I$="S" THEN  GO TO 110
  90 PRINT '"SUSTITUYA EL CONMUT
ADOR DEL"'"ENCENDIDO"
  95 INPUT D$
 100 RUN
 110 PRINT '"IESTAN BIEN LAS CON
EXIONES"'"DE LA BATERIA? (S/N)"
 120 INPUT I$
 130 IF I$="S" THEN  GO TO 160
 140 PRINT '"REPARE LAS CONEXION
ES"
```

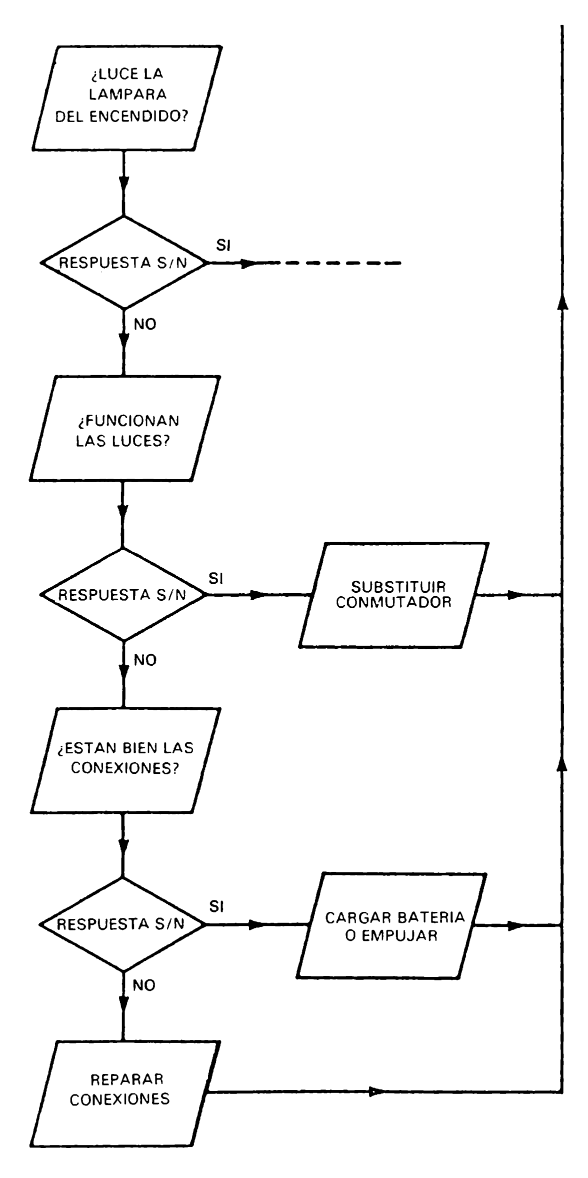


Diagrama de flujo 5.2. Un «experto» encadenado

```
145 INPUT D$
150 RUN
160 PRINT "CARGUE LA BATERIA O
```

```
EMPUJE"
 165 INPUT D$
 170 RUN
 180 PRINT "EL PROGRAMA SEGUIRIA
 CON LOS"'"SINTOMAS MECANICOS"
```

Este tipo de programa es relativamente fácil de redactar pero resulta ineficaz cuando se hace más largo y más complicado.

La utilización de matrices, un medio adecuado para resolver problemas

Una forma más eficaz de tratar la situación es poner el texto en matrices o tablas y disponer de punteros que dirijan a la cuestión o respuesta siguiente según se conteste si o no a la pregunta (véase el Diagrama de flujo 5.3).

El formato para la aplicación de los datos (DATA) para cada caso es:

(TEXTO),(Puntero para el «SI»),(Puntero para el «NO»)

La primera pregunta será:

¿LUCE LA LAMPARA DE IGNICION? (S/N) ... 1

Si la respuesta fuera «N», se precisaría hacer la segunda pregunta:

¿FUNCIONAN LAS LUCES CORRECTAMENTE? (S/N) ... 2

En otro caso se continuaría con otra parte del diagnóstico (que no se ha incluido pero que sería el punto 7).

Tenemos que establecer tres matrices o tablas (arrays): O$(N) contiene el texto de salida, Y(N) el puntero para el «sí», y N(N) el puntero para el «no». Las matrices del sí y el no sólo tienen un carácter pero la longitud (número de caracteres) (L) de la matriz ha de ser suficiente para darles cabida. Para que el programa sea fácil de modificar, se usa la variable NP para el número de puntos. Los datos se

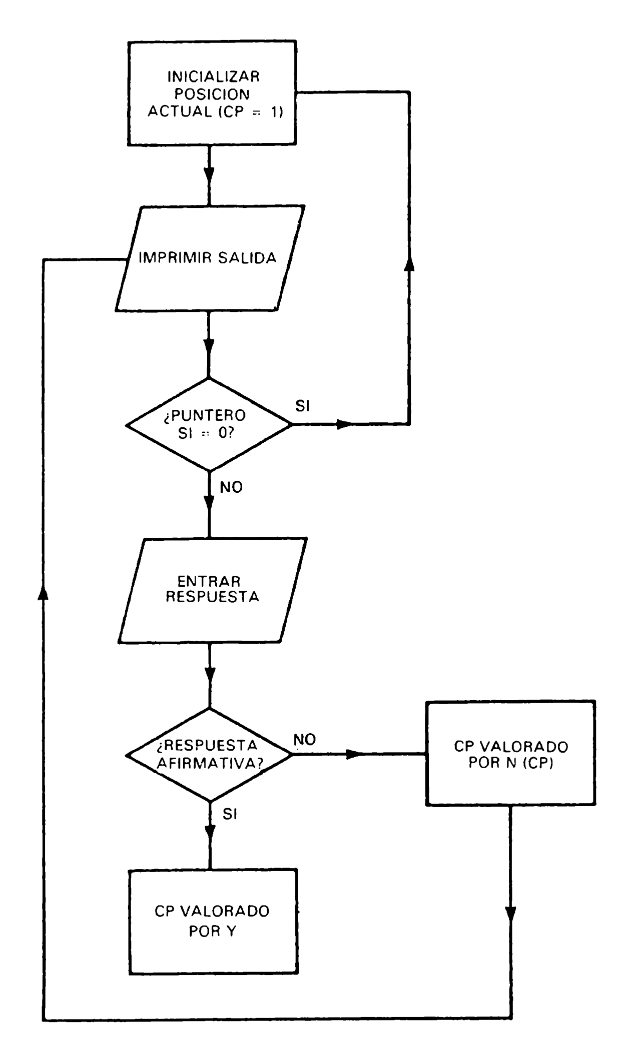


Diagrama de flujo 5.3. Indicación de la salida siguiente

leen en grupos de tres en cada elemento de estas matrices. Cuando el texto de los datos es un posible final del programa se indica poniendo a cero los punteros Y(N) y N(N).

```
    5 RESTORE 5: FOR N=1 TO 7: RE
AD A: POKE USR "I"+N,A: NEXT N:
DATA 2,0,2,14,16,17,14
```

```
  10 GO SUB 8000
8000 LET NP=7: LET L=40
8010 DIM O$(NP,L): DIM Y(NP): DI
M N(NP)
8020 FOR N=1 TO NP
8030 READ O$(N),Y(N),N(N)
8040 NEXT N
8050 RETURN
9000 DATA "ILUCE LA LAMPARA DE I
GNICION?",1,2
9010 DATA "IFUNCIONAN CORRECTAME
NTE LAS LUCES?",3,4
9020 DATA "SUSTITUYA EL CONMUTAD
OR DE ENCENDIDO",0,0
9030 DATA "IESTAN BIEN LAS CONEX
IONES DE LA BATERIA?",5,6
9040 DATA "CARGUE LA BATERIA O E
MPUJE",0,0
9050 DATA "REPARE LAS CONEXIONES
",0,0
9060 DATA "-RESTO DEL PROGRAMA-"
,0,0
```

La verdadera rutina de ejecución es muy sencilla. Se utiliza un puntero CP para indicar la posición de la matriz. Para empezar se pone en 1 y se presenta en pantalla el primer texto. Si se trata de un punto final Y(CP) = 0 (muy poco probable que ocurra al principio), se reinicializa CP en 1 para que empiece nuevamente la secuencia. Por otra parte, si encontramos un puntero no cero, se solicita un INPUT. Si la entrada es un sí («S»), CP adopta el valor contenido en el elemento apropiado de la matriz Y(N), y en otro caso, el de la matriz N(N).

```
 20 LET CP=1
 30 PRINT O$(CP)
 40 IF Y(CP)=0 THEN  GO TO 20
 50 INPUT I$
 55 IF I$="F" THEN  STOP : REM
LINEA PARA TERMINAR SI SE DESEA
```

```
  60 IF I$="S" THEN  LET CP=Y(CP
): GO TO 30
  70 LET CP=N(CP)
  80 GO TO 30
```

La aproximación paralela, un método eficaz

Una alternativa al método de encadenamiento secuencial descrito anteriormente es una aproximación paralela que hace siempre todas las preguntas posibles antes de llegar a una conclusión. Este procedimiento es generalmente más largo que una eficaz estructura en árbol pero es más probable que produzca la respuesta correcta ya que no se omiten puntos de comparación.

Consideremos cómo se podrían distinguir diversas formas de transporte.

Tendremos en cuenta ocho características y pondremos un 1 o un 0 por la presencia o ausencia de cada una de ellas para los cinco modos de transporte de la Tabla 5.1.

Tabla 5.1. Presencia o ausencias de características

	Bicicleta	Automóvil	Tren	Avión	Caballo
Ruedas	1	1	1	1	0
Alas	0	0	0	1	0
Motor	0	1	1	1	0
Neumáticos	1	1	0	1	0
Raíles	0	0	1	0	0
Ventanillas	0	1	1	1	0
Cadenas	1	0	0	0	0
Dirección	1	1	0	1	1

Si se examina detenidamente se observará que el conjunto de resultados varía para cada una de las diferentes posibilidades. Así pues, a través del análisis de las características, podrá establecerse en qué forma varían tales resultados.

Introduciremos estos valores como datos (DATA) y los leeremos mediante la instrucción READ en una matriz bidimensional F(N,N) junto con otra que contenga los nombres de los vehículos O$(N).

```
   5 RESTORE 5: FOR N=1 TO 7: RE
AD A: POKE USR "I"+N,A: NEXT N:
DATA 2,0,2,14,16,17,14
  10 GO SUB 8000
 100 PRINT "ITIENE RUEDAS?"
 500 INPUT "RESPONDA": INPUT I$
 510 LET AN=1: IF I$="N" THEN  L
ET AN=0
 520 FOR N=1 TO 5
 530 IF F(N,1)=AN THEN  PRINT 'O
$(N)
 540 NEXT N
 560 STOP
8000 DIM O$(5,9): DIM F(5,8)
8010 RESTORE 9000
8030 FOR N=1 TO 5
8040 READ O$(N)
8050 FOR M=1 TO 8
8060 READ F(N,M)
8070 NEXT M: NEXT N
8080 RETURN
9000 DATA "BICICLETA",1,0,0,1,0,
0,1,1
9010 DATA "AUTOMOVIL",1,0,1,1,0,
1,0,1
9020 DATA "TREN",1,0,1,0,1,1,0,0
9030 DATA "AVION",1,1,1,1,0,1,0,
1
9040 DATA "CABALLO",0,0,0,0,0,0,
0,1
```

Ahora podemos preguntar si se cumple la primera característica o no, y utilizar la respuesta para presentar las formas de transporte que se adaptan en este determinado momento (véase Diagrama de flujo 5.4).

En este caso, la respuesta «S» nos dará una presentación como la que sigue:

BICICLETA
AUTOMOVIL

TREN
AVION

y la respuesta «N» nos daría sólo:

CABALLO

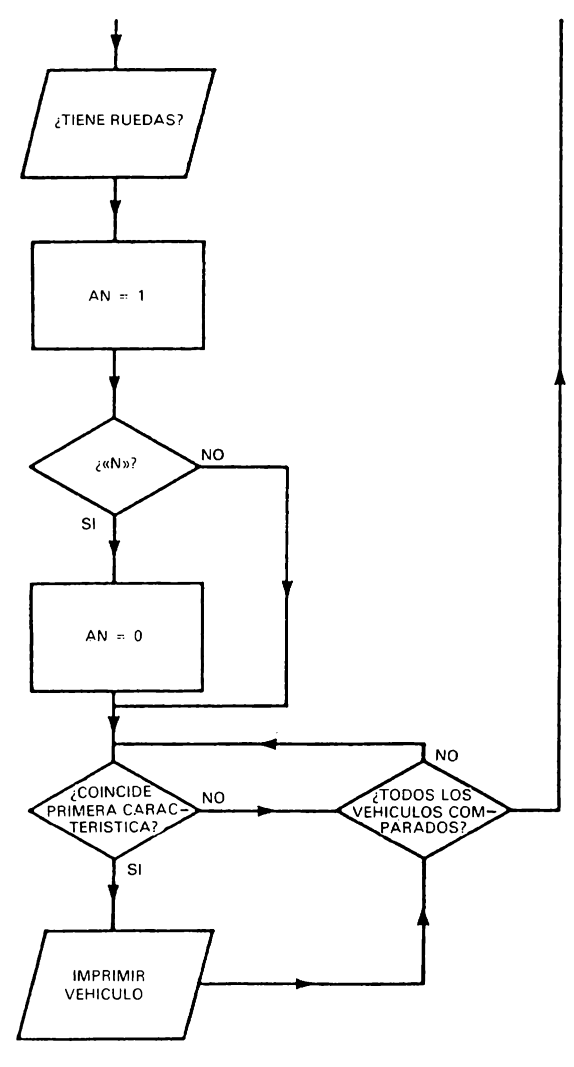


Diagrama de flujo 5.4. Un enfoque paralelo

Esto demuestra claramente una posible desventaja del método paralelo pues, aunque acabamos de ver que sólo el caballo no tiene ruedas, el programa insiste para que continuemos haciendo las siguientes preguntas antes de dar la respuesta. Esto no es tan grave como pudiera parecer a primera vista ya que si se responde «S» a la siguiente interrogación (¿tiene alas?) se verá que el ordenador se niega, muy lógicamente, a creer en caballos con alas.

Si ponemos la parte de la comparación como una subrutina se

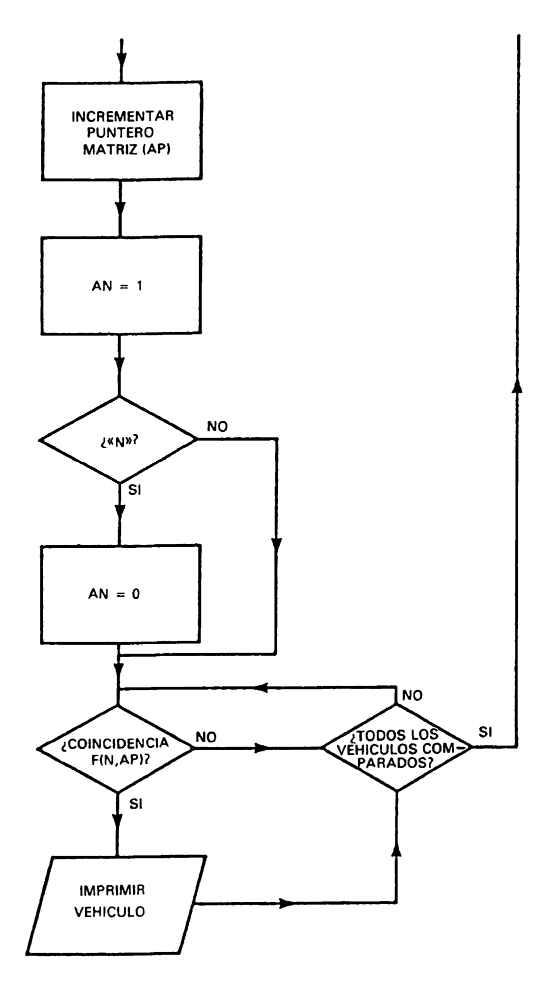


Diagrama de flujo 5.5. Comprobación de las características de una en una

puede utilizar para comprobar sucesivamente cada una de las ocho características. Necesitaríamos ligeras modificaciones: un puntero de matriz AP que se incrementa para hacer la comprobación del siguiente elemento de la matriz F(N,AP) en cada ciclo (véase el Gráfico de flujo 5.5).

```
   5 RESTORE 5: FOR N=1 TO 7: RE
AD A: POKE USR "I"+N,A: NEXT N:
DATA 2,0,2,14,16,17,14
  10 GO SUB 8000
*  90 LET AP=0
 100 PRINT '"ITIENE RUEDAS?"
*110 GO SUB 500
*120 PRINT '"ITIENE ALAS?"
*130 GO SUB 500
*140 PRINT '"ITIENE MOTOR?"
*150 GO SUB 500
*160 PRINT '"ITIENE NEUMATICOS?"
*170 GO SUB 500
*180 PRINT '"INECESITA RAILES?"
*190 GO SUB 500
*200 PRINT '"ITIENE VENTANILLAS?
"
*210 GO SUB 500
*220 PRINT '"ITIENE CADENAS?"
*230 GO SUB 500
*240 PRINT '"ITIENE DIRECCION?"
*250 GO SUB 500
*400 PAUSE 0
*410 RUN
 500 INPUT I$
*510 LET AP=AP+1: LET AN=1: IF I
$="N" THEN  LET AN=0
 520 FOR N=1 TO 5
*530 IF F(N,AP)=AN THEN  PRINT O
$(N)
 540 NEXT N
*550 RETURN
8000 DIM O$(5,9): DIM F(5,8)
8010 RESTORE 9000
8030 FOR N=1 TO 5
```

```
8040 READ O$(N)
8050 FOR M=1 TO 8
8060 READ F(N,M)
8070 NEXT M: NEXT N
8080 RETURN
9000 DATA "BICICLETA",1,0,0,1,0,
0,1,1
9010 DATA "AUTOMOVIL",1,0,1,1,0,
1,0,1
9020 DATA "TREN",1,0,1,0,1,1,0,0
9030 DATA "AVION",1,1,1,1,0,1,0,
1
9040 DATA "CABALLO",0,0,0,0,0,0,
0,1
```

Nota: Las líneas marcadas con un asterisco son las que se incorporan al listado anterior. Ahora puede suprimirse la 560 (STOP) pues con la 400 (PAUSE 0) se puede párar el programa con la instrucción BREAK para salir del bucle sin fin.

La adaptación de las respuestas a los datos por el método de tanteo

La rutina anterior presentará una lista de coincidencias o identidades para cada cuestión a medida que se va desarrollando pero no nos dirá qué conjunto de datos corresponde a la adaptación general de las respuestas a todas las preguntas. Podemos producir un TANTEO que muestre cómo las respuestas se adaptan a los datos por medio de una matriz de acierto S(N) para cada objeto que sólo se incrementa cuando se halla una identidad F(N,AP) = AN (véase el Diagrama de flujo 5.6).

```
   5 RESTORE 5: FOR N=1 TO 7: RE
AD A: POKE USR "I"+N,A: NEXT N:
DATA 2,0,2,14,16,17,14
  10 GO SUB 8000
  90 LET AP=0
 100 PRINT '"ITIENE RUEDAS?"
 110 GO SUB 500
 120 PRINT '"ITIENE ALAS?"
 130 GO SUB 500
 140 PRINT '"ITIENE MOTOR?"
 150 GO SUB 500
 160 PRINT '"ITIENE NEUMATICOS?"
```

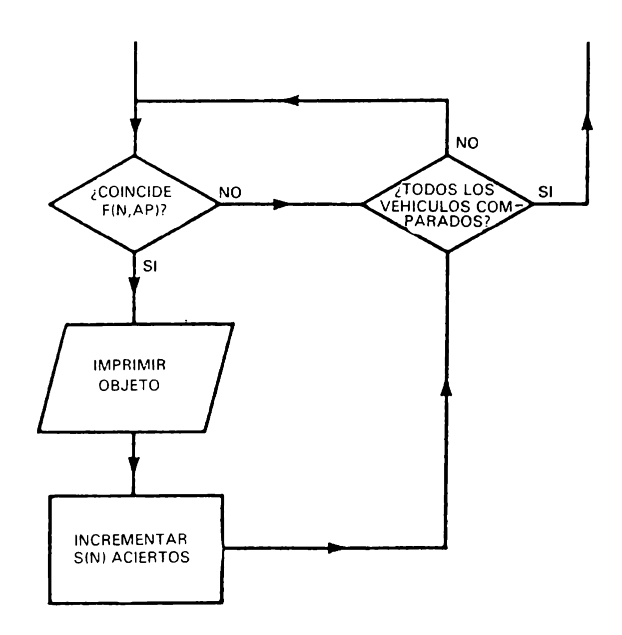


Diagrama de flujo 5.6. Cálculo de las coincidencias

```
 170 GO SUB 500
 180 PRINT '"¿NECESITA RAILES?"
 190 GO SUB 500
 200 PRINT '"¿TIENE VENTANILLAS?
"
 210 GO SUB 500
 220 PRINT '"¿TIENE CADENAS?"
 230 GO SUB 500
 240 PRINT '"¿TIENE DIRECCION?"
 250 GO SUB 500
*260 PRINT
*270 PRINT "PUNTUACION"
*280 PRINT
*300 FOR N=1 TO 5
*310 PRINT O$(N),S(N)
*320 NEXT N
 400 PAUSE 0
 410 RUN
 500 INPUT I$
 510 LET AP=AP+1: LET AN=1: IF I
```

```
$="N" THEN  LET AN=0
 520 FOR N=1 TO 5
*530 IF F(N,AP)=AN THEN  PRINT O
$(N): LET S(N)=S(N)+1
 540 NEXT N
 550 RETURN
 560 STOP
8000 DIM O$(5,9): DIM F(5,8)
*8010 DIM S(5)
8030 FOR N=1 TO 5
8040 READ O$(N)
8050 FOR M=1 TO 8
8060 READ F(N,M)
8070 NEXT M: NEXT N
8080 RETURN
9000 DATA "BICICLETA",1,0,0,1,0,
0,1,1
9010 DATA "AUTOMOVIL",1,0,1,1,0,
1,0,1
9020 DATA "TREN",1,0,1,0,1,1,0,0
9030 DATA "AVION",1,1,1,1,0,1,0,
1
9040 DATA "CABALLO",0,0,0,0,0,0,
0,1
```

Nota: Las líneas marcadas con un asterisco son las que se incorporan en esta fase.

Si se encuentra una adaptación total, S(N) será igual a 8. Cuando uno o más puntos no han sido correctos, el tanteo se reducirá. De esta manera la puntuación por tanteo es particularmente útil cuando las respuestas correctas son más una opinión que un hecho (por ejemplo, ¿tiene un caballo dirección?) ya que la puntuación más alta que se logra señala probablemente la respuesta correcta en cualquier caso. (Obsérvese que en este caso cada respuesta correcta tiene la misma ponderación.)

Un método para ahorrar memoria y tiempo

Se habrá advertido que se han tomado ocho características para la comparación y podrá pensarse que este valor no es accidental ya que

un octeto (byte) tiene ocho bits. Si consideramos cada característica como la representación de un número binario (véase la Tabla 5.2) más que como un valor absoluto, cada objeto podrá describirse como un solo número decimal que es la suma de dígitos binarios en lugar de ocho valores separados. Haremos la conversión a decimal con el bit menos significativo en la parte superior por lo que, al empezar por arriba con las ruedas, cada característica es equivalente a 1,2,4,8,16, 32,64,128 en notación decimal, respectivamente.

Tabla 5.2. Características expresadas por su valoración binaria

	Bicicleta	Automóvil	Tren	Avión	Caballo
Ruedas	1	1	1	1	0
Alas	0	0	0	2	0
Motor	0	4	4	4	0
Neumáticos	8	8	0	8	0
Raíles	0	0	16	0	0
Ventanillas	0	32	32	32	0
Cadenas	64	0	0	0	0
Dirección	128	128	0	128	128
Suma total	201	173	53	175	128

No es difícil convertir nuestra puntuación o tanteo de 1 a 8 en el correspondiente valor binario siempre que recordemos que el valor decimal del dígito binario —BV— debe doblarse cada vez que nos desplazamos hacia abajo, y que sólo necesitamos añadir el valor binario de la puntuación en cada momento si la respuesta es afirmativa («S») (AN = 1, véase el Diagrama de flujo 5.7).

Si se piensa un momento, se advertirá que sólo necesitamos analizar el número total producido —S— agregando los valores binarios de las respuestas afirmativas. No es preciso hacer un bucle y analizar cada parte del contenido de la matriz cada vez, ni siquiera hace falta disponer de una matriz bidimensional. Los únicos datos que necesitamos entrar son los valores decimales de cada objeto, D(N), y cuando se han hecho todas las preguntas se comparan aquellos valores con los decimales obtenidos por la conversión bi-

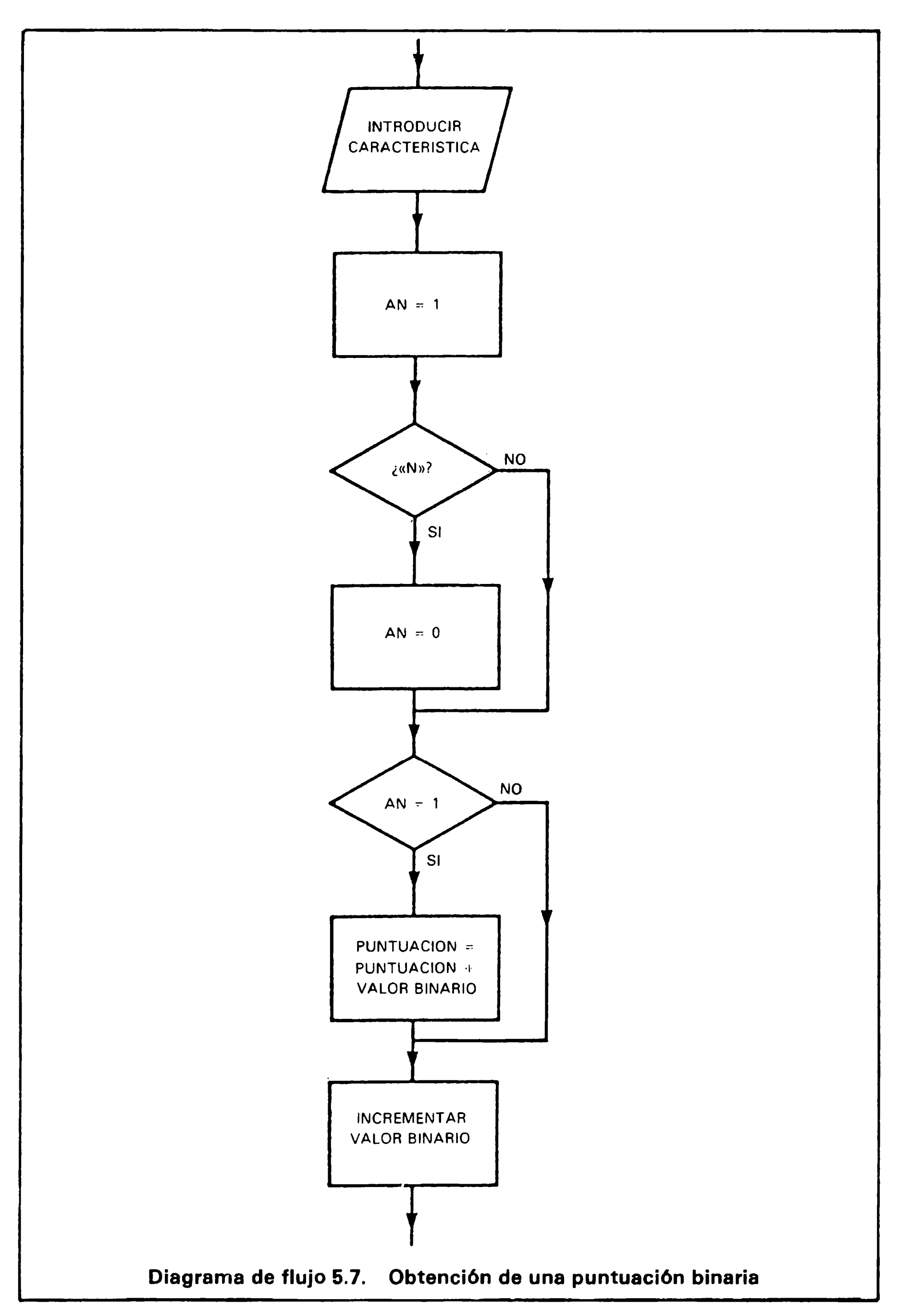


Diagrama de flujo 5.7. Obtención de una puntuación binaria

```
   5 RESTORE 5: FOR N=1 TO 7: RE
AD A: POKE USR "I"+N,A: NEXT N:
DATA 2,0,2,14,16,17,14
  10 GO SUB 8000
  90 LET AP=0
 100 PRINT '"ITIENE RUEDAS?"
 110 GO SUB 500
 120 PRINT '"ITIENE ALAS?"
 130 GO SUB 500
 140 PRINT '"ITIENE MOTOR?"
 150 GO SUB 500
 160 PRINT '"ITIENE NEUMATICOS?"
 170 GO SUB 500
 180 PRINT '"INECESITA RAILES?"
 190 GO SUB 500
 200 PRINT '"ITIENE VENTANILLAS?
"
 210 GO SUB 500
 220 PRINT '"ITIENE CADENAS?"
 230 GO SUB 500
 240 PRINT '"ITIENE DIRECCION?"
 250 GO SUB 500
 260 PRINT
 270 PRINT "PUNTUACION ";SU
 280 PRINT
 300 FOR N=1 TO 5
*310 IF D(N)=SU THEN  PRINT ,O$(
N): GO TO 400
 320 NEXT N
*330 PRINT "VEHICULO NO ENCONTRA
DO"
*400 PAUSE 0: STOP
 410 INPUT I$: RUN
 500 INPUT I$
*510 LET AN=1: IF I$="N" THEN  L
ET AN=0
*520 IF AN=1 THEN  LET SU=SU+BV
 530 LET BV=BV+BV
 550 RETURN
*8000 DIM O$(5,9): DIM D(5): LET
BV=1: LET SU=0
```

```
*9000 DATA "BICICLETA",201
*9010 DATA "AUTOMOVIL",173
*9020 DATA "TREN",53
*9030 DATA "AVION",175
*9040 DATA "CABALLO",128
*9050 FOR N=1 TO 5
*9060 READ O$(N),D(N)
 9070 NEXT N
 9080 RETURN
```

Nota: Las líneas marcadas con un asterisco son las que se incorporan en esta fase.

naria de las respuestas «sí/no», SU (véase el Diagrama de flujo 5.8). Lo mejor ahora es borrar todo lo que sigue a la línea 260 y empezar a programar de nuevo.

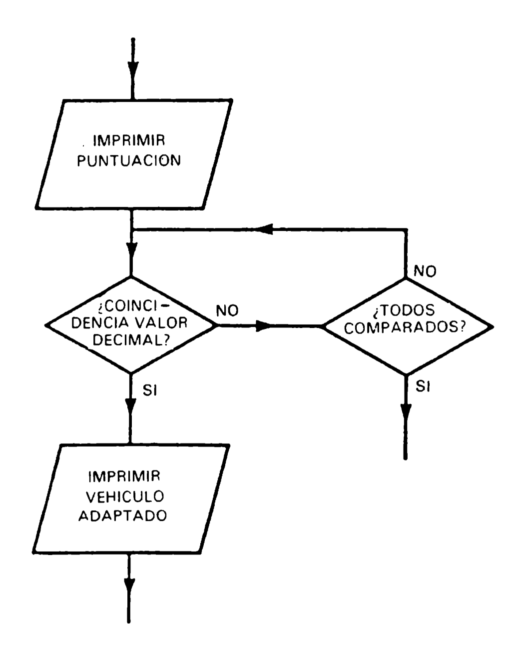


Diagrama de flujo 5.8. Coincidencia con valores decimales

Este enfoque del problema ahorra obviamente gran cantidad de memoria y tiempo ya que cada elemento de la matriz tiene varios oc-

tetos (bytes) y debe localizarse antes de hacer la comparación, por lo que resulta extremadamente útil cuando se trabaja con grandes cantidades de información. Por otra parte, esto quiere decir que hay que calcular los equivalentes decimales de cada caso antes de poder emplearlos y tampoco nos da pista alguna cuando no se logra la coincidencia o identidad completa. (Téngase en cuenta que no es posible redondear los valores decimales ya que el valor equivalente de cada respuesta correcta depende de su posición.)

Por supuesto, que los cálculos pueden hacerse por el lado difícil pero el Spectrum tiene una función binaria (BIN) por lo que basta poner en la instrucción PRINT una serie de dígitos binarios con el comando BIN. Ejemplo:

```
PRINT BIN 10101010
```

que nos dará el valor decimal 170. (No se olvide que la tabla se lee de arriba hacia abajo por lo que el octeto o byte menos significativo se lee el primero.)

6

El autoaprendizaje de un sistema experto

Aunque los sistemas expertos que hemos descrito hasta ahora funcionan perfectamente, todos requieren que se les dé anticipadamente las reglas correctas en las que basar sus decisiones y esto puede ser muy tedioso.

Es posible, sin embargo, preparar un programa experto que aprenda de sus propios errores y establezca por sí mismo sus reglas de decisión siempre que pueda decirse cuándo se equivoca (aunque no dónde). Esto es, desde luego, una ventaja si uno no está muy seguro de cuáles son tales reglas correctas. En este caso empezamos con una serie de características que nos permitan distinguir entre diferentes objetos, pero sin un modelo predefinido de síes y de noes (regla de decisión) relativo a tales características, que pudiera guiarnos. En su lugar, utilizamos el propio programa para calcular el citado modelo.

Trabajaremos con nuestro conocido ejemplo de medios de transporte y comenzaremos por establecer algunas variables. FE es el número de características a considerar (8); F$(N) es una matriz para contener sus nombres; F(N) contendrá los valores que se dan a cada característica como entrada en un momento determinado (0 ó 1); y R(N) mantendrá los valores actuales de la regla de decisión de cada rasgo.

```
   5 RESTORE 5: FOR N=1 TO 7: RE
AD A: POKE USR "I"+N,A: NEXT N:
DATA 2,0,2,14,16,17,14
```

```
  10 GO SUB 8000
8000 LET FE=8
8010 DIM F$(FE,11): DIM F(FE): D
IM R(FE)
8020 FOR N=1 TO FE
8030 READ F$(N)
8035 PRINT F$(N)
8040 NEXT N
9000 DATA "RUEDAS","ALAS","MOTOR
","NEUMATICOS","RAILES","VENTANI
LLAS","CADENA","DIRECCION"
9010 RETURN
```

Cada característica se considera separadamente (véase el Diagrama de flujo 6.1). Primero, para este ciclo, se pone a cero el valor de la característica —F(N)— y después se solicita una entrada de un «Sí/No» en la variable I$. Si I$ es «S», el elemento del valor del F(N) se pone a 1; en otro caso, permanece en 0. Esto producirá un modelo que describe el objeto con ceros y unos en la matriz F(N).

```
 60 FOR N=1 TO FE
 70 LET F(N)=0
 80 PRINT F$(N);"  ";
 90 PAUSE 0
100 INPUT "IS/N?";I$: PRINT I$,
110 IF I$="S" THEN  LET F(N)=1
120 NEXT N
```

Ahora la variable de decisión DE se pone a cero y se vuelve a calcular como la suma del valor presente de DE más cada uno de los valores del elemento F(N) que se ha entrado multiplicados por los valores actuales de la regla de decisión R(N).

```
125 LET DE=0
130 FOR N=1 TO FE
150 LET DE=DE+F(N)*R(N)
160 NEXT N
170 PRINT "DE= ";DE
```

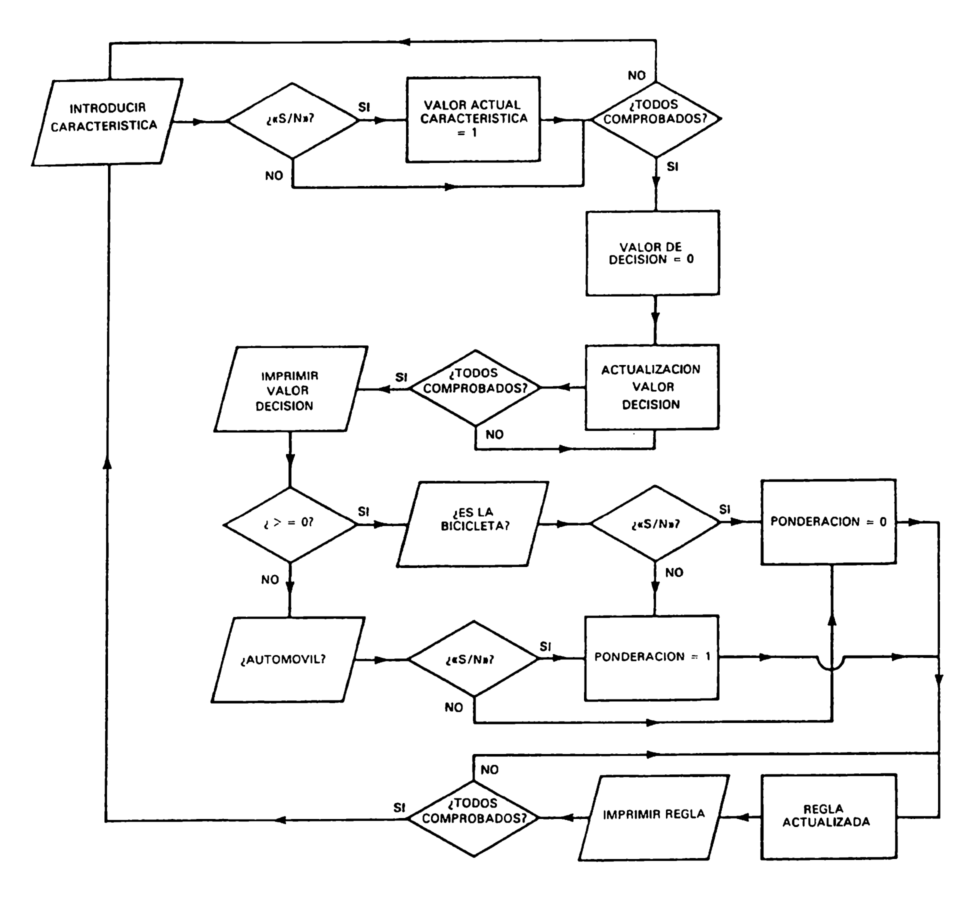


Diagrama de flujo 6.1. Aprendizaje del método para distinguir entre dos objetos

La distinción entre dos objetos y la autocorrección del sistema

Para empezar consideraremos la situación más sencilla en la que haya sólo dos posibilidades: una bicicleta o un coche. Inicialmente hacemos la distinción entre ellos arbitrariamente. Establecemos que si el valor final de DE es igual o mayor que cero se trata de una bicicleta; si es menor que cero, el vehículo es un coche. No importa que esto no sea verdaderamente cierto ya que el sistema se autocorregirá. Cuando el programa ha tomado una decisión sobre la base del valor de DE solicita confirmación del resultado.

```
 180 IF DE>=0 THEN  PRINT "IES U
NA BICICLETA? ";: INPUT "IS/N?";
I$: PRINT I$: GO TO 200
 190 IF DE<0 THEN  PRINT "IES UN
 COCHE? ": INPUT I$: PRINT I$: G
O TO 220
```

Se puede actuar de forma distinta según que la decisión del ordenador haya sido correcta o no. En caso afirmativo, no se realiza ninguna acción (una variable de ponderación WT se pone a cero) y el programa vuelve a probar otra vez. Si DE ha sido mayor o igual a cero, pero la respuesta era incorrecta, WT se pone a menos uno, mientras que si DE era menor que cero y la respuesta igualmente equivocada, WT se pone a más uno.

```
 200 IF I$="S" THEN  LET WT=0: G
O TO 240
 210 LET WT=-1: GO TO 240
 220 IF I$="S" THEN  LET WT=0: G
O TO 240
 230 LET WT=1
```

El efecto de la variable de ponderación es el de modificar los valores de la matriz R(N), reduciéndolos cuando son demasiado altos y aumentándolos si son muy bajos.

```
240 FOR N=1 TO FE
250 LET R(N)=R(N)+F(N)*WT
260 PRINT R(N),
270 NEXT N
280 PRINT : PRINT
290 GO TO 60
```

La forma en que trabaja el sistema se verá mejor mediante una demostración. Ejecútese el programa (RUN) y sígase esta secuencia de entradas. (Obsérvese que los separadores se han puesto de tal forma que producen un formato de pantalla señalando claramente la relación entre los valores de entrada y los valores de la regla de decisión.)

Introdúzcanse estos valores:

RUEDAS	S	ALAS	N
MOTOR	N	NEUMATICOS	S
RAILES	N	VENTANILLAS	N
CADENA	S	DIRECCION	S

El programa nos dará un valor de decisión DE igual a cero que es el inicial pues aún no se han producido modificaciones:

DE = 0

Puesto que DE es 0, el sistema supone que se trata de una bicicleta y pide confirmación, a la que corresponde naturalmente la respuesta «Sí».

¿ES UNA BICICLETA? S

La presentación en pantalla del contenido de la matriz de control R(N) muestra que sus valores continúan siendo cero al producir la respuesta correcta por puro azar.

0	0
0	0
0	0
0	0

Pruébese ahora a entrar esta secuencia que describe un automóvil:

RUEDAS	S	ALAS	N
MOTOR	S	NEUMATICOS	S
RAILES	N	VENTANILLAS	S
CADENA	N	DIRECCION	S

DE sigue siendo cero. Así pues se ha logrado una conclusión errónea y se hace la pregunta equivocada, a la que hay que responder con un «No».

DE = 0
¿ES UNA BICICLETA? N

Ahora, como se ha cometido una equivocación, la regla de decisión se modifica restando uno de cada valor de la matriz de control en que se respondió con un «Sí». El contenido de tal matriz queda así:

```
1     0
1    —1
0    —1
0    —1
```

Si se aplican nuevamente los valores que describen un coche, el programa llegará a una respuesta correcta:

```
RUEDAS        S     ALAS           N
MOTOR         S     NEUMATICOS     S
RAILES        N     VENTANILLAS    S
CADENA        N     DIRECCIÓN      S
DE = —5
¿ES UN COCHE?     S
1     0
1    —1
0    —1
0    —1
```

Antes de sentirnos demasiado satisfechos probemos a entrar nuevamente los valores para una bicicleta. ¡Respuesta equivocada!

```
RUEDAS        S     ALAS           N
MOTOR         N     NEUMATICOS     S
RAILES        N     VENTANILLAS    N
CADENA        S     DIRECCION      S
DE = —3
¿ES UN COCHE?     N
0     0
1     0
0    —1
1     0
```

Sin embargo, las características positivas que son comunes a la bicicleta y al automóvil se incrementan ahora automáticamente en uno de forma que si nuevamente se repite esta última secuencia se tendrá una conclusión correcta.

```
RUEDAS          S       ALAS           N
MOTOR           N       NEUMATICOS     S
RAILES          N       VENTANILLAS    N
CADENA          S       DIRECCION      S
DE = 1
¿ES UNA  BICICLETA?       S
  0                       0
—1                        0
  0                      —1
  1                       0
```

La situación se ha estabilizado ahora y el programa ya reconocerá siempre correctamente un coche y una bicicleta cada vez que se entren las características que los definen:

```
RUEDAS          S       ALAS           N
MOTOR           S       NEUMATICOS     S
RAILES          N       VENTANILLAS    S
CADENA          N       DIRECCION      S
DE = —2
¿ES UN COCHE?       S
  0                       0
—1                        0
  0                      —1
  1                       0
```

Obsérvese que el valor final de DE para la bicicleta es 1 y para el automóvil —2. Si se miran los valores de la matriz de control se advertirá que éstos se corresponden en número y posición con las características únicas que distinguen estos objetos (CADENA para la bicicleta y MOTOR y VENTANILLAS para el coche).

Ampliación del número de posibilidades que puede tratar un ordenador

Aunque ya hemos logrado enseñar al ordenador algo, no es para echar las campanas al vuelo pues sólo ha aprendido a distinguir entre dos objetos. Expandamos el sistema para permitirle tratar con un espectro mayor de posibilidades (véase el Diagrama de flujo 6.2). Para empezar necesitamos definir el número de objetos que deseamos que reconozca —OB—, ponerles nombre en instrucciones DATA que podamos leer, mediante la instrucción READ e incluir en una nueva matriz O$(OB), cambiar nuestra matriz de control de decisión a una forma bidimensional, R(FE,OB), que pueda albergar reglas para cada uno de los objetos separadamente, y establecer una matriz de decisión, D(N), que retenga valores de decisión para cada objeto.

```
  10 GO SUB 8000
8000 LET FE=8: LET OB=5
8010 DIM F$(FE,11): DIM F(FE): D
IM R(FE,OB): DIM O$(OB,8): DIM D
(OB): DIM Q$(8,1)
8015 RESTORE 9000
8020 FOR N=1 TO FE
8030 READ F$(N)
8040 NEXT N
8050 FOR N=1 TO OB
8060 READ O$(N)
8070 NEXT N
9000 DATA "RUEDAS","ALAS","MOTOR
","NEUMATICOS","RAILES","VENTANI
LLAS","CADENA","DIRECCION"
9010 DATA "BICICL.","AUTOMOV.","
TREN","AVION","CABALLO"
9999 RETURN
```

En lugar de tener una sola variable de decisión DE, necesitamos para determinar un valor de decisión para cada objeto cada vez. En cada ciclo debemos poner primero DE a cero y después dar igualmente este valor a cada elemento de la matriz de decisión D(N), para empezar a cero con cada objeto.

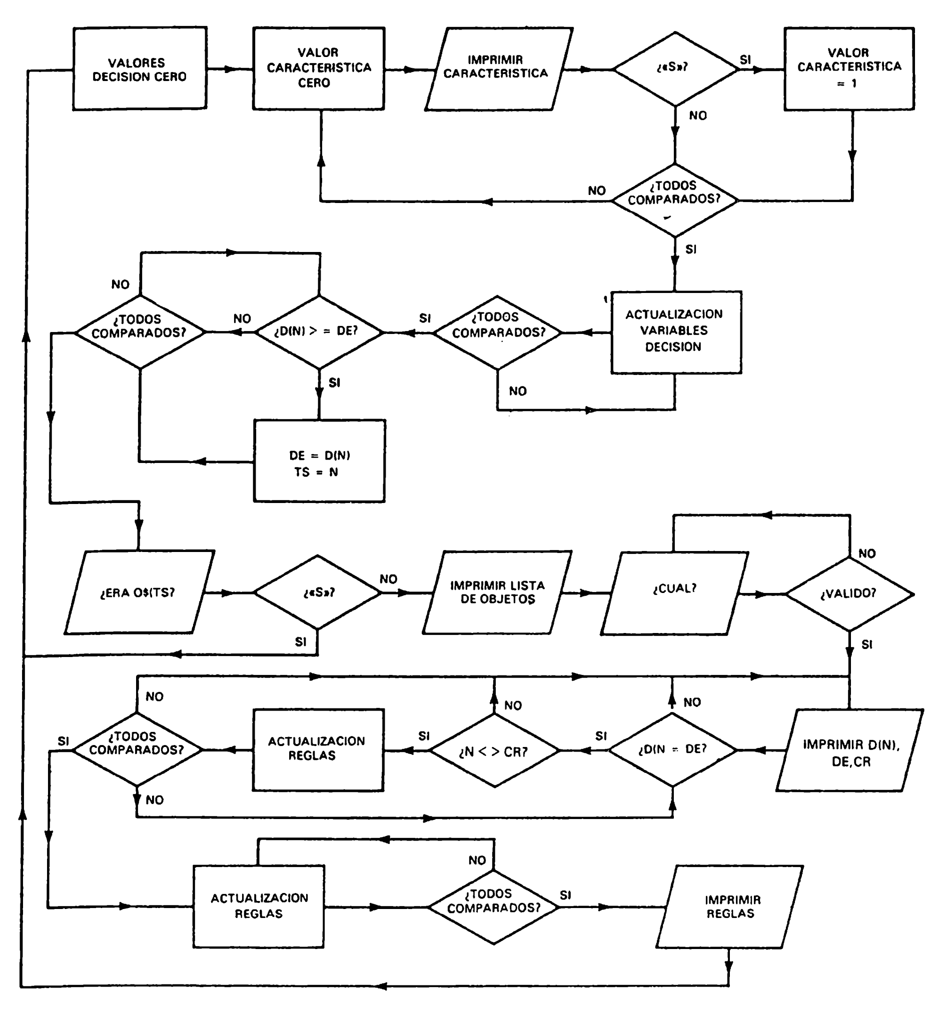


Diagrama de flujo 6.2. **Aprendizaje de las reglas para un espectro más amplio de posibilidades.**

```
20 LET DE=0
30 FOR N=1 TO OB
40 LET D(N)=0
50 NEXT N
```

Los valores para cada característica se entran exactamente igual que antes.

```
 60 FOR N=1 TO FE
 70 LET F(N)=0
 80 PRINT F$(N);" ";
 90 INPUT "IS/N?";I$
100 PRINT I$,
110 IF I$="S" OR I$="s" THEN  L
ET F(N)=1
120 NEXT N
```

Cada elemento de la matriz de decisión D(N), se actualiza ahora de acuerdo con el estado de los valores F(N) entrados y el contenido del apropiado elemento de la ordenación de control R(N,M).

```
130 FOR N=1 TO FE
140 FOR M=1 TO OB
150 LET D(M)=D(M)+F(N)*R(N,M)
160 NEXT M: NEXT N
```

Ahora necesitamos mirar si cualesquiera de los valores de decisión para todos los objetos son mayores o iguales al valor total de decisión DE. En este caso, hacemos una variable de puntuación máxima TS igual al número que produce la mejor adaptación, N.

```
170 FOR N=1 TO OB
180 IF D(N)>=DE THEN  LET DE=D(
N): LET TS=N
190 NEXT N
```

Lo mejor que puede adivinar el sistema es que esta es la respuesta correcta. Así que pide nuevamente confirmación y simplemente regresa a una nueva entrada sin hacer ningún cambio si la respuesta fue correcta.

```
200 PRINT "IERA ";O$(TS);"?";
210 INPUT "IS/N?";I$: PRINT I$
220 IF I$="S" OR I$="s" THEN  G
O TO 20
```

Si esta no ha sido la respuesta correcta, los nombres y números de todos los objetos son presentados en pantalla y se pide el número de la respuesta acertada CR. (Las limitaciones de esta variable impiden que el programa se interrumpa por la aplicación de un valor inadmisible).

```
 230 FOR N=1 TO OB
 240 PRINT N;"  ";O$(N),
 250 NEXT N: PRINT
 260 PRINT "ICUAL ERA?   ";
 270 INPUT CR: IF CR<1 OR CR>5 T
HEN  GO TO 270
 275 PRINT CR
```

Se comprueba ahora si el valor de decisión para cada objeto D(N) es mayor o igual al total DE y si además el objeto considerado no es la respuesta correcta. Si ambas cosas son ciertas se vuelven a actualizar las reglas mediante la reducción de los valores de las características correctas F(N) para hacerlos tender hacia el de la respuesta acertada.

```
 280 PRINT TAB (8);"D(N)    DE
  CR": FOR N=1 TO OB
 290 PRINT TAB (9);D(N);TAB (16)
;DE;TAB (22);CR
 300 IF D(N)>=DE AND N<>CR THEN
 FOR M=1 TO FE: LET R(M,N)=R(M,N
)-F(M): NEXT M
 310 NEXT N
```

Ahora los valores de característica correctos F(N) se suman a la matriz de control para el objeto correcto para hacer que la tendencia vaya en sentido opuesto.

```
320 FOR M=1 TO FE
330 LET R(M,CR)=R(M,CR)+F(M)
340 NEXT M: PRINT
```

Finalmente se presentan en pantalla los valores de las matrices de control para que pueda verse lo que está sucediendo.

```
 350 FOR M=1 TO OB
 360 FOR N=1 TO FE
 370 PRINT TAB (N*3)-3;R(N,M);
 380 NEXT N
 390 PRINT TAB (24);O$(M)
 400 NEXT M
 410 RESTORE 460
 420 FOR N=1 TO FE
 430 READ Q$(N)
 440 PRINT TAB (N*3)-3;Q$(N);
 450 NEXT N
 460 DATA "A","B","C","D","E","F
","G","H"
 465 PRINT ''"PULSE UNA TECLA PA
RA SEGUIR Y LA F PARA PARAR"
 470 INPUT W$: IF W$="F" THEN  S
TOP
 480 CLS : GO TO 20
```

Lo mejor es hacer nuevamente una demostración para comprender lo que sucede. Así que apliquemos la siguiente secuencia:

RUEDAS	S	ALAS	N
MOTOR	N	NEUMATICOS	S
RAILES	N	VENTANILLAS	N
CADENA	S	DIRECCION	S

El programa llegará a la conclusión errónea de que se trataba de un caballo por lo que hay que decirle que está equivocado cuando nos pida la respuesta correcta (bicicleta = 1):

```
¿ERA EL CABALLO?        N
1  BICICLETA      2  AUTOMOVIL
3  TREN           4  AVION
5  CABALLO
¿CUAL ERA?        1
```

La situación de las diversas matrices de decisión y de control se presentan a continuación:

D(N)	DE	CR
0	0	1
0	0	1
0	0	1
0	0	1
0	0	1

1	0	0	1	0	0	1	1	bicicleta
—1	0	0	—1	0	0	—1	—1	automóvil
—1	0	0	—1	0	0	—1	—1	tren
—1	0	0	—1	0	0	—1	—1	avión
—1	0	0	—1	0	0	—1	—1	caballo
A	B	C	D	E	F	G	H	

(A = ruedas B = alas C = motor D = neumáticos
E = raíles F = ventanillas G = cadena H = dirección)

Si se mira atentamente se observará que las características que han causado alteraciones en las matrices de control son ruedas, neumáticos, cadena y dirección, que son los mismos que definimos como partes de una bicicleta pero que no se encuentran en un caballo. Además, se verá que los valores de estas características para la bicicleta son ahora más uno, mientras que los de los otros objetos toman el valor de menos uno.

Ahora damos las características de un coche, con las que el ordenador cree que es una bicicleta y los corrige. Obsérvese que la matriz de control para bicicleta y automóvil es enmendada ahora al considerar la nueva información.

```
RUEDAS          S      ALAS           N
MOTOR           S      NEUMATICOS     S
RAILES          N      VENTANILLAS    S
CADENA          N      DIRECCION      S
¿ERA BICICLETA?     N
1  BICICLETA     2  AUTOMOVIL
```

3 TREN 4 AVION
5 CABALLO
¿CUAL ERA? 2

D(N)	DE	CR
3	3	2
—3	3	2
—3	3	2
—3	3	2
—3	3	2

0	0	—1	0	0	—1	1	0	bicicleta
0	0	1	0	0	1	—1	0	automóvil
—1	0	0	—1	0	0	—1	—1	tren
—1	0	0	—1	0	0	—1	—1	avión
—1	0	0	—1	0	0	—1	—1	caballo
A	B	C	D	E	F	G	H	

A continuación se dan las de un avión y decide que es un automóvil, con nueva corrección.

RUEDAS S ALAS S
MOTOR S NEUMATICOS S
RAILES N VENTANILLAS S
CADENA N DIRECCION S
¿ERA AUTOMOVIL? N
1 BICICLETA 2 AUTOMOVIL
3 TREN 4 AVION
5 CABALLO
¿CUAL ERA? 4

Y ahora las características del tren, y nueva respuesta errónea.

RUEDAS S ALAS N
MOTOR S NEUMATICOS N
RAILES S VENTANILLAS S

CADENA	N	DIRECCION	N

¿ERA UN AVION? N

1	BICICLETA	2	AUTOMOVIL
3	TREN	4	AVION
5	CABALLO		

¿CUAL ERA? 3

Y finalmente las del caballo, respondiendo que avión.

RUEDAS	N	ALAS	N
MOTOR	N	NEUMATICOS	N
RAILES	N	VENTANILLAS	N
CADENA	N	DIRECCION	S

¿ERA UN AVION? N

1	BICICLETA	2	AUTOMOVIL
3	TREN	4	AVION
5	CABALLO		

¿CUAL ERA? 5

La incorporación automática de la información para un sistema experto

Si continuamos aplicando información a nuestro experto conseguiremos que finalmente nos responda correctamente cada vez. El tiempo que precise dependerá de la extensión de las diferencias entre las características de los objetos y en el orden con que se presentan al experto. Téngase en cuenta que puede pasar mucho tiempo hasta lograr que sea infalible. He aquí una secuencia que finalmente se logró correctamente cada vez:

avión (tren)	automóvil (avión)	bicicleta (SI)
automóvil (SI)	avión (automóvil)	avión (SI)
caballo (SI)	avión (bicicleta)	automóvil (avión)
avión (automóvil)	avión (automóvil)	automóvil (avión)
automóvil (SI)	avión (automóvil)	avión (SI)
automóvil (SI)	avión (SI)	caballo (SI)
bicicleta (SI)	tren (automóvil)	tren (SI)

bicicleta (SI)	automóvil (avión)	automóvil (SI)
avión (automóvil)	avión (SI)	automóvil (avión)
automóvil (SI)	avión (SI)	automóvil (SI)
bicicleta (automóvil)	automóvil (SI)	avión (SI)
tren (SI)	caballo (SI)	bicicleta (SI)

Para ver el estado final de la matriz de control cuando se ha llegado a él, se puede detener el programa y ordenarle «GO TO 350» como orden directa. Como la escala final de valores varía entre +6 y —2 no deberá sorprender el largo tiempo que se requiere para llegar allí.

1	0	—1	1	0	—2	3	0	(bicicleta)
—1	4	1	0	—1	1	—2	0	(automóvil)
0	—1	1	—2	2	1	—1	—2	(tren)
—2	6	0	—1	—1	0	—2	—2	(avión)
—1	0	0	—1	0	0	—1	0	(caballo)
A	B	C	D	E	F	G	H	

(A = ruedas	B = alas	C = motor	D = neumáticos
E = raíles	F = ventanillas	G = cadena	H = dirección)

Por supuesto que en una aplicación real de este sistema experto sería posible incorporar automáticamente una masa de información sobre el tema en cuestión, así como las conclusiones, dejando sólo al ordenador para que asimilase dicha información y lograse las reglas oportunas en tiempo adecuado. Como tales reglas se almacenan en matrices se podría preparar fácilmente una rutina para conservarlas y poder usarlas posteriormente.

(Nota del Adaptador: Se han incorporado al listado las líneas 465 y 470 con objeto de poder salir del programa cuando nos cansemos.)

A continuación se presenta el listado completo del programa:

```
   5 RESTORE 5: FOR N=1 TO 7: RE
AD A: POKE USR "I"+N,A: NEXT N:
DATA 2,0,2,14,16,17,14
  10 GO SUB 8000
  20 LET DE=0
```

```
  30 FOR N=1 TO OB
  40 LET D(N)=0
  50 NEXT N
  60 FOR N=1 TO FE
  70 LET F(N)=0
  80 PRINT F$(N);" ";
  90 INPUT "IS/N?";I$
 100 PRINT I$,
 110 IF I$="S" OR I$="s" THEN  L
ET F(N)=1
 120 NEXT N
 130 FOR N=1 TO FE
 140 FOR M=1 TO OB
 150 LET D(M)=D(M)+F(N)*R(N,M)
 160 NEXT M: NEXT N
 170 FOR N=1 TO OB
 180 IF D(N)>=DE THEN  LET DE=D(
N): LET TS=N
 190 NEXT N
 200 PRINT "IERA ";O$(TS);"?";
 210 INPUT "IS/N?";I$: PRINT I$
 220 IF I$="S" OR I$="s" THEN  G
O TO 20
 230 FOR N=1 TO OB
 240 PRINT N;" ";O$(N),
 250 NEXT N: PRINT
 260 PRINT "ICUAL ERA?   ";
 270 INPUT CR: IF CR<1 OR CR>5 T
HEN  GO TO 270
 275 PRINT CR
 280 PRINT TAB (8);"D(N)     DE
  CR": FOR N=1 TO OB
 290 PRINT TAB (9);D(N);TAB (16)
;DE;TAB (22);CR
 300 IF D(N)>=DE AND N<>CR THEN
 FOR M=1 TO FE: LET R(M,N)=R(M,N
)-F(M): NEXT M
 310 NEXT N
 320 FOR M=1 TO FE
 330 LET R(M,CR)=R(M,CR)+F(M)
```

```
 340 NEXT M: PRINT
 350 FOR M=1 TO OB
 360 FOR N=1 TO FE
 370 PRINT TAB (N*3)-3;R(N,M);
 380 NEXT N
 390 PRINT TAB (24);O$(M)
 400 NEXT M
 410 RESTORE 460
 420 FOR N=1 TO FE
 430 READ Q$(N)
 440 PRINT TAB (N*3)-3;Q$(N);
 450 NEXT N
 460 DATA "A","B","C","D","E","F
","G","H"
 465 PRINT ''"PULSE UNA TECLA PA
RA SEGUIR Y LA F PARA PARAR"
 470 INPUT W$: IF W$="F" THEN  S
TOP
 480 CLS : GO TO 20
8000 LET FE=8: LET OB=5
8010 DIM F$(FE,11): DIM F(FE): D
IM R(FE,OB): DIM O$(OB,8): DIM D
(OB): DIM Q$(8,1)
8015 RESTORE 9000
8020 FOR N=1 TO FE
8030 READ F$(N)
8040 NEXT N
8050 FOR N=1 TO OB
8060 READ O$(N)
8070 NEXT N
9000 DATA "RUEDAS","ALAS","MOTOR
","NEUMATICOS","RAILES","VENTANI
LLAS","CADENA","DIRECCION"
9010 DATA "BICICL.","AUTOMOV.","
TREN","AVION","CABALLO"
9999 RETURN
```

7

La recuperación de la información de ficheros

Los ordenadores son totalmente lógicos pero nuestros propios bancos de memoria no lo son tanto, dando lugar a ciertos problemas cuando se trata de recuperar información sobre un cierto tema. Por ejemplo, el idioma inglés es una lengua muy variable y con frecuencia presenta nombres iguales (o muy similares) escritos de forma diferente, con las dificultades consiguientes. Una solución a este problema es tratar de identificar el sonido de la palabra, en lugar de las letras con que se escribe, mediante el «Soundex Coding» (Código Sonoro) que se desarrolló en Estados Unidos para ayudar a realizar el Censo de 1890. Este método de codificación asegura que palabras de sonido similar tienen casi el mismo código secuencial.

Las reglas para codificar una palabra son las siguientes:

1) Siempre se conserva la primera letra de la palabra como el primer carácter del código.

Desde la siguiente letra en adelante:

2) Se ignoran las vocales (a, e, i, o, u).
3) Se ignoran las letras *w, y, q* y *h.*
4) Se ignoran igualmente los signos de puntuación.
5) Se codifican las otras letras con valores del 1 al 6 en la forma siguiente:

NOTA DEL TRADUCTOR/ADAPTADOR: Este Capítulo se incluye como ejemplo del desarrollo de las técnicas cuyos principios se exponen en esta obra. Como puede comprobar el lector, se trata de un ejercicio práctico que necesariamente se ha de desarrollar utilizando las palabras inglesas para la identificación de los sonidos de acuerdo con el método de codificación «Soundex Coding». Lo importante aquí es deducir las aplicaciones que pueden derivarse de los principios expuestos.

Letras	Código
bfpv	1
cgjksxz	2
dt	3
l	4
mn	5
r	6

6) Cuando letras contiguas tienen el mismo código sólo se retiene la primera.
7) Si el código tiene más de cuatro caracteres sólo se tomarán los cuatro primeros.
8) Si el código tiene menos de cuatro caracteres se completará hasta este número con ceros.

Para aclarar lo que antecede se presentan a continuación algunos ejemplos de nombres codificados por el sistema «Soundex».

BRAIN — B650

(La B se conserva, R es 6, A e I se ignoran, N es 5 y se añade un cero para completar el código.)

CUNNINGHAM — C552

(C se conserva; U se ignora; las dos N se representan por un solo código, el 5; I se ignora; la tercera N es 5; G es 2; H y A se ignoran; y M es 5. Pero el código resultante (C5525) se corta para dejarlo con cuatro caracteres.)

GORE — G600

(La G se conserva, la O se ignora, R es 6, E se ignora y se añaden ceros para completar el código.)

IRELAND — I645

(La I se conserva, la R es 6, la E se ignora, L es 4, A se ignora, N es 5 y D es 3. El código resultante (I6453) se corta hasta dejarlo con cuatro caracteres.)

SCOT — S300

(La S se conserva, la C se ignora porque está en el mismo grupo que la S, la O se ignora, T es 3 y se añaden ceros hasta cuatro caracteres.)

Si la palabra está llena de vocales y de otras letras que se desechan nos encontraremos con un código verdaderamente breve.

HEYHOE — H000

(La H se retiene, las demás letras se rechazan y se completa el código con ceros.)

Rutina de Codificación

Para informatizar el código elaboremos un programa que nos permita entrar una palabra y obtener su código «Soundex» (véase el Diagrama de flujo 7.1). Lo primero a realizar es establecer algunas instrucciones DATA que contengan las letras retenidas en sus correspondientes grupos. (Obsérvese que tales grupos se ordenan de acuerdo con el valor del código.)

```
9000 DATA "BFPV","CGJKSXZ","DT",
"L","MN","R"
```

Ahora podemos entrar la palabra a codificar —I$— y, para empezar, hacer que C$ sea su primera letra (regla 1).

```
100 RESTORE : INPUT I$
110 LET C$=I$(1)
```

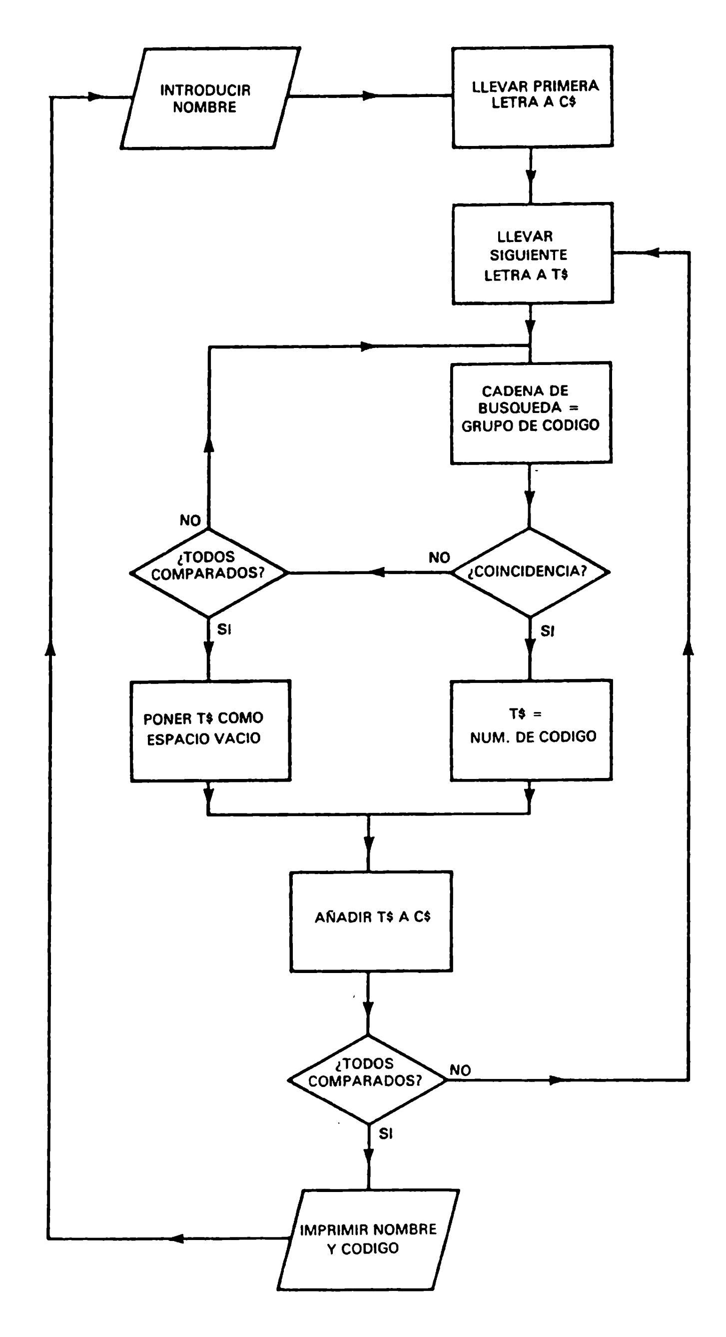


Diagrama de flujo 7.1. Creación del Código «Soundex»

Ahora necesitamos comprobar las otras letras de la palabra, 2 TO LEN (I$), una por una, después de hacer una cadena temporal T$ para poner la letra a analizar.

```
 120 FOR N=2 TO LEN (I$): RESTOR
E
 130 LET T$=I$(N)
```

Como la conversión de los números de código se precisará en varios puntos del problema final, estableceremos este proceso como una subrutina en la línea 1000.

```
140 GO SUB 1000
```

Tenemos que comparar T$ con cada letra de los grupos. Para la comprobación de cada grupo hemos de repetir el ciclo seis veces, haciendo una cadena de análisis S$ para el código «Soundex» de cada uno de tales grupos, y usar una rutina INSTR (véase Cap. 2) que compare cada una de las letras del grupo con la que se investiga en T$.

```
1000 FOR P=1 TO 6
1010 READ S$
1020 GO SUB 5000
```

La rutina INSTR es similar a la utilizada en los capítulos anteriores.

```
5000 FOR M=1 TO LEN (S$)
5010 IF S$(M)=T$ THEN  LET SP=M:
 RETURN
5020 NEXT M
5030 LET SP=0
5040 RETURN
```

Cuando se efectúa la comparación INSTR, se ha de determinar si ha habido identidad o coincidencia con alguno de los grupos

«Soundex» y en este caso, con cuál. En caso negativo, SP se pone a cero y si hay identificación adopta el valor de M que corresponde al del grupo del código adaptado.

Cuando hay identidad o coincidencia (SP>0), convertimos el valor del bucle que explora los grupos P del código en una cadena T$ que substituye la que teníamos con carácter temporal.

```
1030 IF SP>0 THEN  LET T$=STR$ (
P): RETURN
```

Si no hay identidad o coincidencia en ese grupo hay que comprobar el siguiente.

```
1040 NEXT P
```

Y si después de todo el análisis, sigue sin haberla, T$ debe contener uno de los caracteres que hay que ignorar. Así, pues, hacemos T$ una cadena vacía (T$ = " ") y regresamos al programa (RETURN).

```
1050 LET T$=""
1060 RETURN
```

Ahora podemos hacer la cadena codificada C$ igual a la original codificada, más el carácter recientemente convertido T$.

```
170 LET C$=C$+T$
180 NEXT N
```

Al llegar a este punto hemos de volver atrás para tratar el siguiente carácter en I$.

Cuando se alcanza el fin de I$, se presenta (PRINT) la entrada (I$) y la tòtalidad de la cadena codificada C$ antes de volver a la línea 100 para una nueva palabra.

```
 210 PRINT : PRINT "NOMBRE","COD
IGO": PRINT I$,C$
 320 GO TO 100
```

Si se aplica el nombre STEVEN se producirá el código S315 que es correcto. Si, por otra parte, se prueba con BRAIN y CUNNINGHAM se tendrán B65 y C55525 respectivamente. El de BRAIN es demasiado corto y necesita ampliarse con ceros y el de CUNNINGHAM es muy largo. Se repiten los mismos códigos uno tras otro para la letra N.

Resolución de casos especiales: Códigos demasiado cortos o cadenas largas

Para solucionar el problema de la repetición del mismo código para letras contiguas se necesita un registro de la última cadena temporal: L$. Tenemos que hacer que L$ sea el código del primer carácter de I$ para empezar, de forma que la letra inicial no se repita. Según se va realizando el bucle FOR-NEXT necesitamos comparar L$ con T$, y si son iguales, no se añadirá T$ a C$. En otro caso, es preciso hacer que L$ sea el último valor de T$.

```
 110 LET T$=I$(1): LET C$=T$: GO
SUB 1000: LET L$=T$
 150 IF T$=L$ THEN  GO TO 180
 160 LET L$=T$
```

Ahora podemos resolver el problema de código demasiado corto. En primer lugar, se comprueba la longitud de la cadena, LEN (C$)<4. Si es este el caso, se añaden tres ceros y después se corta la cadena en su tamaño correcto (cuatro caracteres).

```
 190 IF LEN (C$)<4 THEN  LET C$=
C$+"000": LET C$=C$( TO 4)
```

Finalmente si la cadena es demasiado larga, se corta nuevamente para dejarla en su longitud correcta.

```
 200 IF LEN (C$)>4 THEN  LET C$=
C$( TO 4)
```

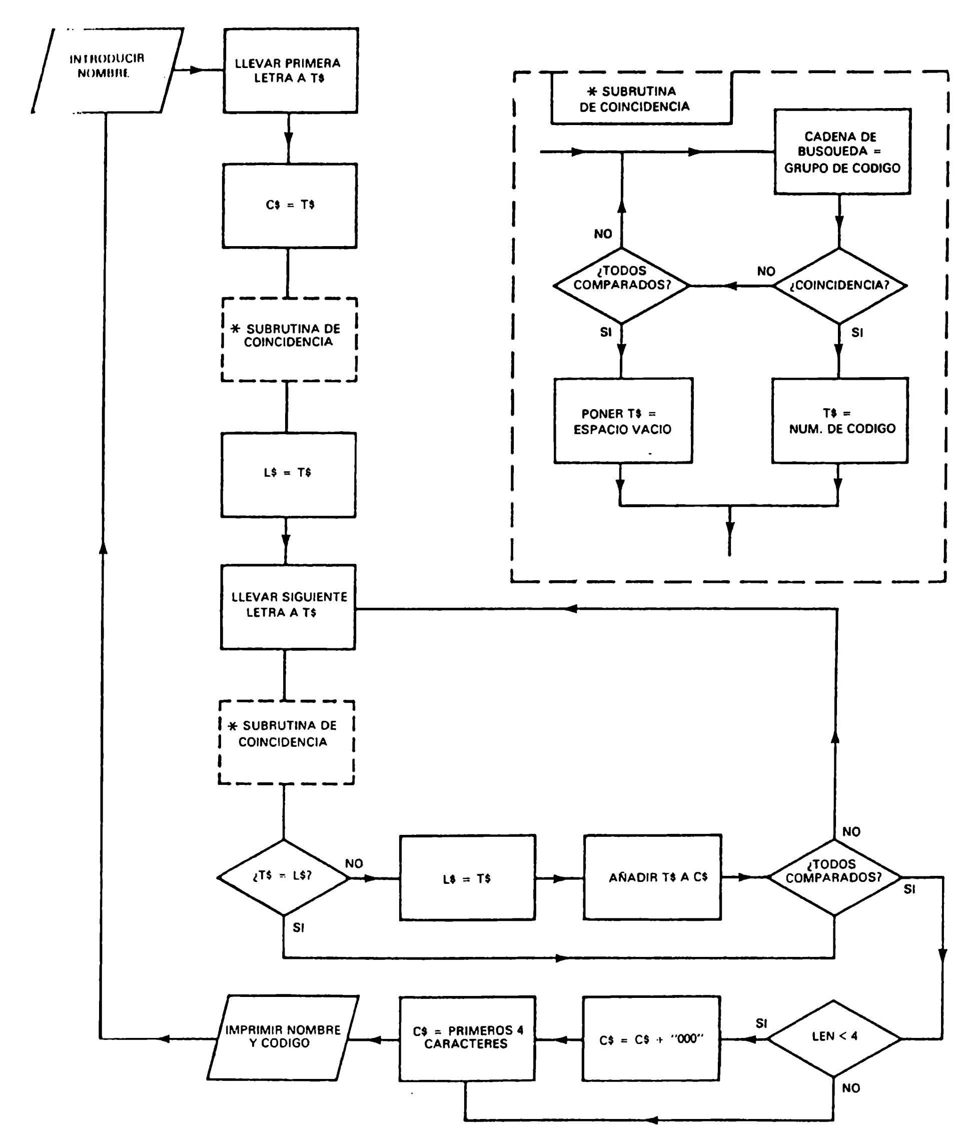


Diagrama de flujo 7.2. La creación de detalles

Aplicaciones de codificación

Ahora que tenemos un método útil para producir el Código «Soundex» démosle algo con que trabajar. La primera tarea es prepa-

rar una lista de nombres en declaraciones DATA y haremos que se lean y apliquen a una matriz en cadena, N$(N). Nuestra lista de demostración se compone de dieciocho nombres. Si se desean más, bastará una rápida ojeada al listín telefónico local para resolver el problema. Adviértase que el número de palabras se almacena también en la variable NW.

```
  10 GO SUB 8000
8000 LET NW=18
8010 DIM N$(NW,9)
9010 DATA "ABRAHAM","ABRAHAMS","
ABRAMS","ADAM","ADAMS","ADDAMS",
"ADAMSON","ALAN","ALLAN","ALLEN"
9020 DATA "ANTHANY","ANTHONY","A
NTONY","ANTROBUS","APPERLEY","AP
PLEBEE","APPLEBY","APPLEFORD"
9030 RESTORE 9010: FOR N=1 TO NW
9040 READ N$(N)
9050 NEXT N
```

La idea que se persigue con la codificación «Soundex» descansa en el hecho de que se use la adaptación antes de presentar las palabras posibles. Tenemos, por consiguiente, que encontrar los códigos de cada nombre de las declaraciones DATA y poner estos códigos en una matriz en cadena equivalente, O$(N). La rutina para hallar el Código «Soundex» es virtualmente idéntica a la usada para encontrar el código de una palabra entrada con una instrucción INPUT, como se describió anteriormente.

```
9500 DIM O$(NW,4)
9510 PRINT : PRINT "NOMBRE","COD
IGO": PRINT
9520 FOR Q=1 TO NW
9530 PRINT N$(Q),
9540 LET T$=N$(Q)( TO 1): LET C$
=T$: RESTORE : GO SUB 1000: LET
L$=T$
9550 FOR N=2 TO LEN (N$(Q))
```

```
9560 LET T$=N$(Q)(N)
9570 RESTORE : GO SUB 1000
9580 IF T$=L$ THEN  NEXT N: GO T
O 9620
9590 LET L$=T$
9600 LET C$=C$+T$
9610 NEXT N
9620 IF LEN (C$)<4 THEN  LET C$=
C$+"000": LET C$=C$( TO 4)
9630 IF LEN (C$)>4 THEN  LET C$=
C$( TO 4)
9640 PRINT C$
9650 LET O$(Q)=C$
9660 NEXT Q
9670 RETURN
```

Si se ejecuta (RUN) esto ahora se verán todos los códigos de las palabras introducidas en las declaraciones DATA antes de que se pida la entrada de una palabra.

NOMBRE	CODIGO
ABRAHAM	A165
ABRAHAMS	A165
ABRAMS	A165
ADAM	A350
ADAMS	A352
ADDAMS	A352
ADAMSON	A352
ALAN	A450
ALLAN	A450
ALLEN	A450
ANTHANY	A535
ANTHONY	A535
ANTONY	A535
ANTROBUS	A536
APPERLEY	A164
APPLEBEE	A141

APPLEBY	A141
APPLEFORD	A141

Lo único que necesitamos en este momento es encontrar qué códigos de estos nombres igualan al de la entrada y después presentar estos nombres por medio de un bucle FOR-NEXT.

```
 240 PRINT
 250 FOR N=1 TO NW
 260 IF C$=O$(N) THEN  PRINT N$(
N),O$(N)
 270 NEXT N
```

Con esto sólo se presentarán palabras con idénticos Códigos «Soundex». Por ejemplo, si se prueba a entrar el nombre APPLEBE se tendrá la siguiente respuesta:

?APPLEBE

NOMBRE	CODIGO
APPLEBE	A141
APPLEBEE	A141
APPLEBY	A141
APPLEFORD	A141

Aunque APPLEBE (con una E al final) no se halla en los datos (DATA), se han encontrado APPLEBEE y APPLEBY así como APPLEFORD (donde se ha cortado el final de la palabra).

Adaptaciones parciales de la codificación

Obśervese que, sin embargo, se ha rechazado APPERLEY aunque tiene una pronunciación muy similar (en inglés, N. del T.). Sería útil, por consiguiente, que se pudieran hacer adaptaciones parciales.

Esto se puede conseguir fácilmente añadiendo otro bucle FOR-NEXT que compare una sección decreciente de la entrada con longitudes igualmente decrecientes de los códigos almacenados (véase el Diagrama de flujo 7.3).

```
 230>FOR M=4TO 1STEP -1
 240 PRINT : PRINT M;" ADAPTACIO
N DE CARACTERES": PRINT
 260 IF C$( TO M)=O$(N)( TO M) T
HEN  PRINT N$(N),O$(N)
 280 PRINT : PRINT "PULSE UNA TE
CLA PARA SEGUIR"
 290 PAUSE 0
 300 PRINT : PRINT
 310 NEXT M
```

Si ahora se prueba APPLEBE podrán verse todas las posibilidades que brinda el programa.

?APPLEBE

NOMBRE	CODIGO
APPLEBE	A141

ADAPTACION CON 4 CARACTERES

APPLEBEE	A141
APPLEBY	A141
APPLEFORD	A141

PULSE UNA TECLA PARA SEGUIR

ADAPTACION CON 3 CARACTERES

APPLEBEE	A141
APPLEBY	A141
APPLEFORD	A141

PULSE UNA TECLA PARA SEGUIR

ADAPTACION CON 2 CARACTERES

ABRAHAM	A165

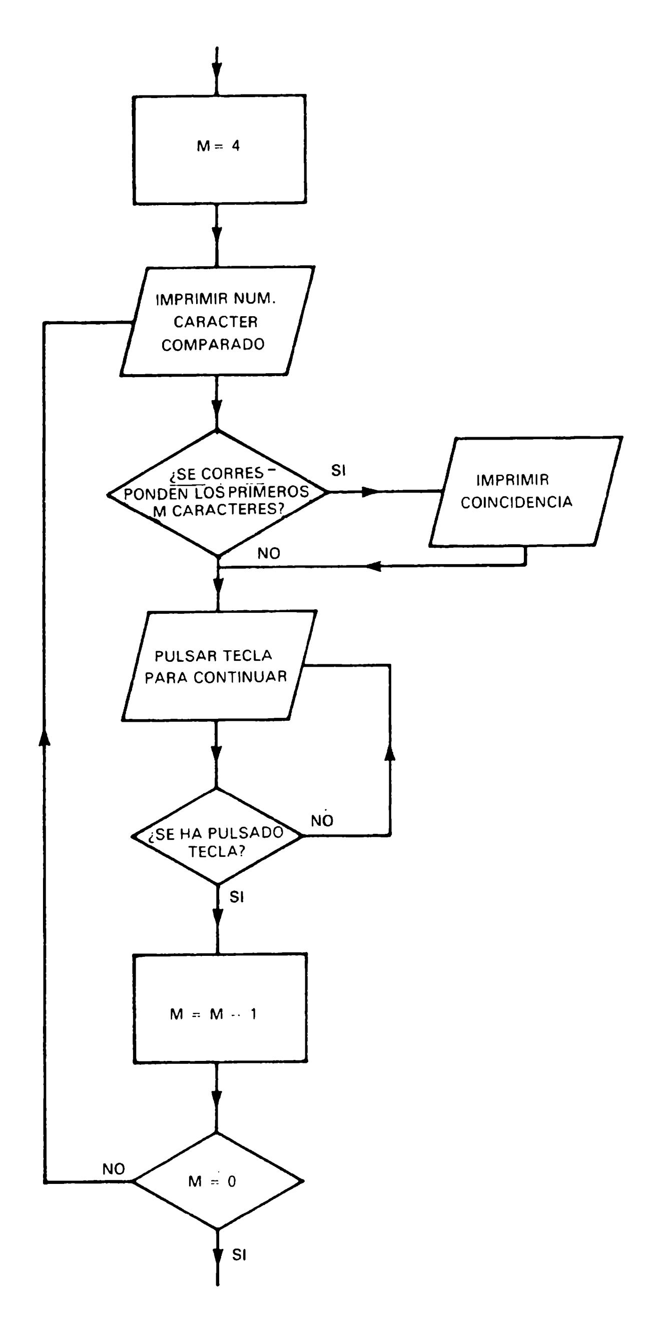


Diagrama de flujo 7.3. Adaptación parcial

```
ABRAHAMS        A165
ABRAMS          A165
APPERLEY        A164
APPLEBEE        A141
APPLEBY         A141
APPLEFORD       A141
PULSE UNA TECLA PARA SEGUIR

ADAPTACION CON 1 CARACTER
ABRAHAM         A165
ABRAHAMS        A165
ABRAMS          A165
ADAM            A350
ADAMS           A352
ADDAMS          A352
ADAMSON         A352
ALAN            A450
ALLAN           A450
ALLEN           A450
ANTHANY         A535
ANTHONY         A535
ANTONY          A535
ANTROBUS        A536
APPERLEY        A164
APPLEBEE        A141
APPLEBY         A141
APPLEFORD       A141
PULSE UNA TECLA PARA SEGUIR
```

8

El reconocimiento de las formas gráficas

Normalmente reconocemos los objetos mediante los sentidos de la vista, el oído, el gusto y el tacto, mientras que nuestro ordenador sólo puede lograr información a través del teclado. Si bien es posible producir sensores que se apliquen a la máquina para proporcionar otro punto de vista del mundo exterior, su construcción requiere una cierta complicación electrónica y mecánica. Vamos a simular la acción de un sensor óptico para ilustrar cómo pueden reconocerse las formas gráficas.

Consideremos para empezar tres formas simples: una línea vertical, un cuadrado y un triángulo rectángulo.

Pueden reconocerse estas formas observando el modelo que organizan sobre una cuadrícula imaginaria y comprobando si existe, o no, un punto de la figura en cada uno de los de la cuadrícula definidos por sus coordenadas X e Y.

En el caso de la línea sólo se utiliza la primera abscisa y todos los valores de las ordenadas. El cuadrado es un poco más complicado ya que todas las coordenadas X de las filas 1 y 8 de las Y están activadas y entre las filas 2 a 7 de las Y sólo están activados el primero y el último puntos de las X. Finalmente, el triángulo es aún más complicado, ya que la inclinación de la hipotenusa se produce incrementando el valor de X en cada salto (Fig. 8.0).

1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1
1	0	0	0	0	1	0	1
1	0	0	0	1	0	0	1
1	0	0	1	0	0	0	1
1	0	1	0	0	0	0	1
1	1	0	0	0	0	0	1
1	1	1	1	1	1	1	1

Figura 8.0. Transformación de gráficos en valores numéricos

Una forma sencilla de describir estas figuras sería la de representar cada punto por un dígito binario y producir el valor decimal del binario resultante del análisis de la serie de dígitos de la fila, de manera análoga a como se hizo en el caso de los sistemas expertos (véase la Tabla 8.1). En realidad, este tipo de estudio se emplea para producir los caracteres que se ven en la presentación de pantalla cuyos formatos se almacenan en la memoria justamente de esta forma. Por ejemplo, la figura 8.1 muestra cómo se compone la letra «A».

Existen ahora máquinas (Lectores Opticos de Caracteres) que pueden invertir este proceso. Realmente «leen» una página impresa explorando el papel según un modelo en forma de cuadrícula y determinando si la luz se refleja, o no, en cada punto definido por coordenadas.

Tabla 8.1. Valores decimales de las formas descritas en dígitos binarios.

Fila	Línea	Cuadrado	Triángulo
1	1	255	1
2	1	129	3
3	1	129	5
4	1	129	9
5	1	129	17
6	1	129	33
7	1	129	65
8	1	255	255

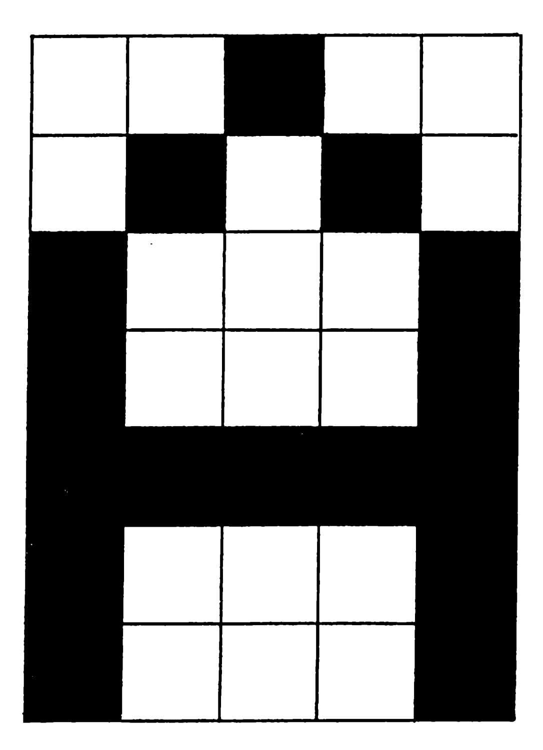


Figura 8.1. Formación de la letra «A»

Lo que verdaderamente se toma es un conjunto de «síes» o «noes» para cada coordenada y luego se decodifica y compara con los modelos de las formas conocidas. La forma más sencilla de hacer esta comparación sería la de considerar cada punto como un dígito binario y convertir la fila de ellos en el valor decimal correspondiente que se compara con una tabla de valores conocidos. Esto tiene, sin embargo, la desventaja de que debe analizarse cada punto en un conjunto de 64 (los que componen una cuadrícula de ocho por ocho).

Método abreviado de análisis de gráficos

Un enfoque más rápido se basa en el hecho de que cada carácter puede detectarse observando solamente un número mucho más limitado de características críticas del modelo. La figura 8.2 nos da un ejemplo de un árbol de decisión que permite encontrar todas las letras

mayúsculas del alfabeto utilizando solamente 12 puntos (véase la Fig. 8.3) y sin que ni siquiera sea necesario analizarlos todos en cada caso. Si se siguen cada una de las rutas se verá que el número máximo de pasos a seguir es siete y que la mayoría de las letras se encuentran en menos de cinco (Tabla 8.2). No cabe duda que esto es mucho más rápido que comparar 64 puntos.

Tabla 8.2. Número de pasos requeridos para el reconocimiento de cada carácter

3 pasos:	I, D
4 pasos:	L, J, C, G, O, W
5 pasos:	S, A, Q, R, T, F, U, espacio
6 pasos:	P, V, Y, H
7 pasos:	B, M, N, E, K, X, Z

Para demostrar cómo funciona este método simularemos la acción de una cabeza exploradora sobre una cuadrícula en la pantalla en la que puedan representarse caracteres.

Se borra la pantalla y se establece una zona negra (PAPER 0) de 6 × 8 bloques en la parte superior izquierda de la pantalla. A dicha zona se superpone otra amarilla (PAPER 6) de 5 × 7 para marcar la de trabajo (debe dejarse naturalmente un margen alrededor para que los caracteres no se mezclen).

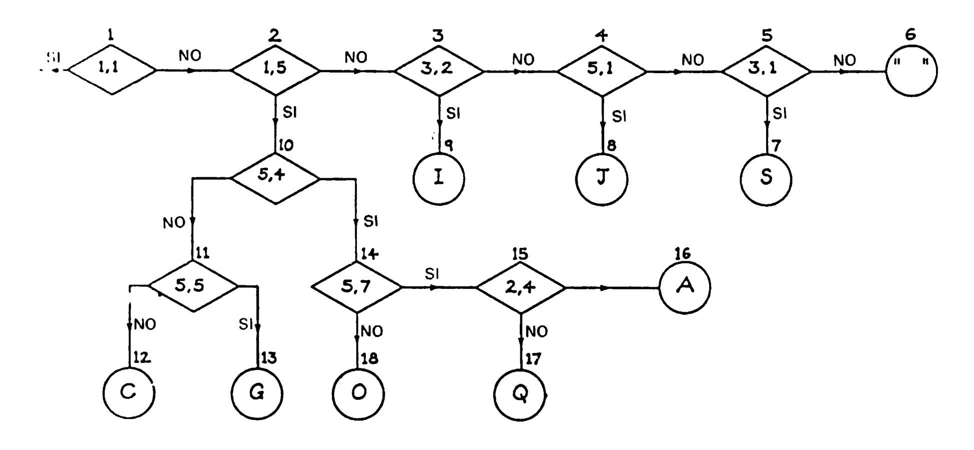
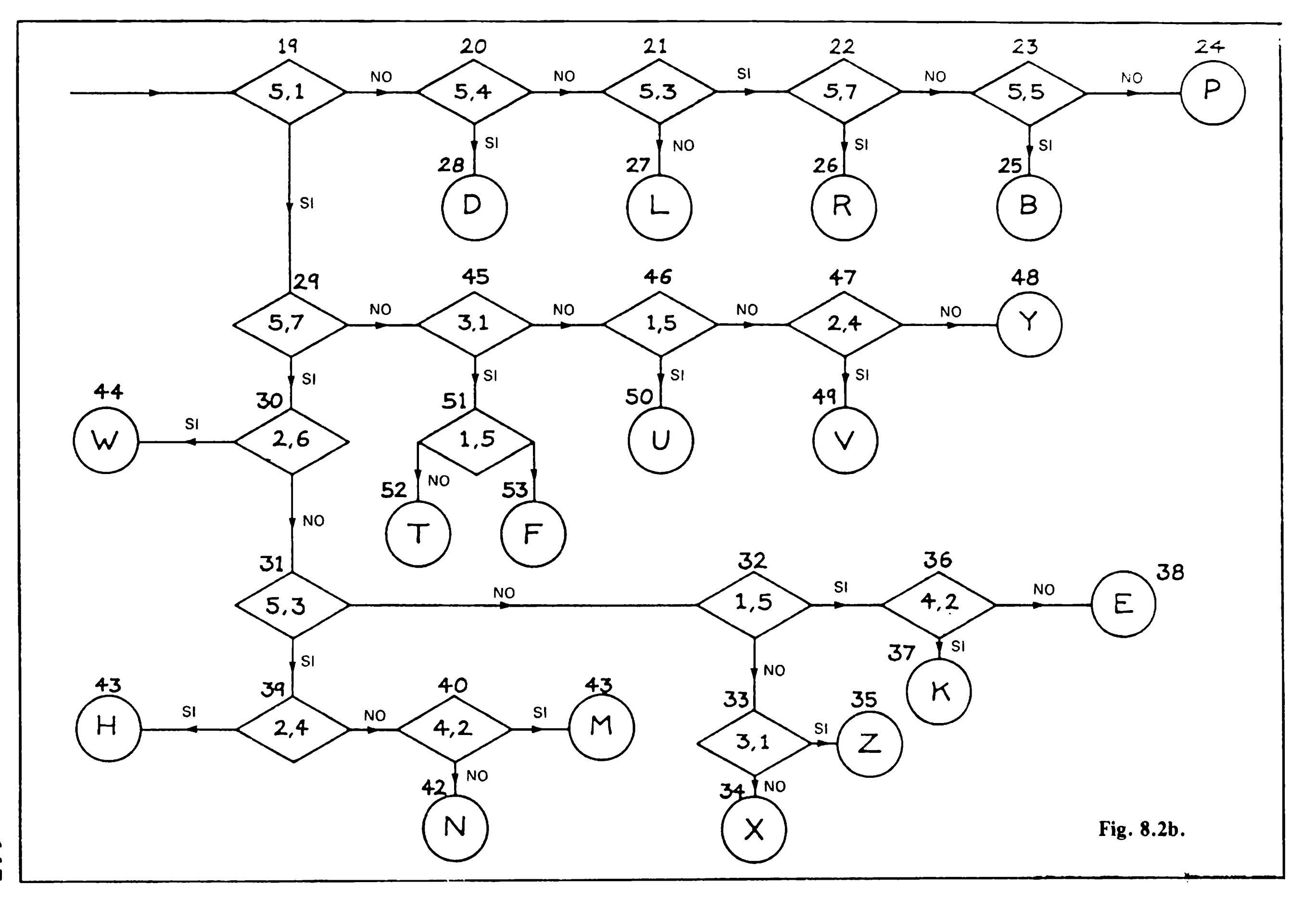


Figura 8.2A. Arbol de decisión para el alfabeto



19
5,1
20
5,4
28
D
21
5,3
27
L
22
5,7
26
R
23
5,5
25
B
24
P
29
5,7
45
3,1
46
1,5
47
2,4
48
Y
50
U
49
V
44
W
30
2,6
51
1,5
52
T
53
F
31
5,3
32
1,5
36
4,2
38
E
37
K
43
H
39
2,4
40
4,2
43
M
42
N
33
3,1
35
Z
34
X
SI
NO

Fig. 8.2b.

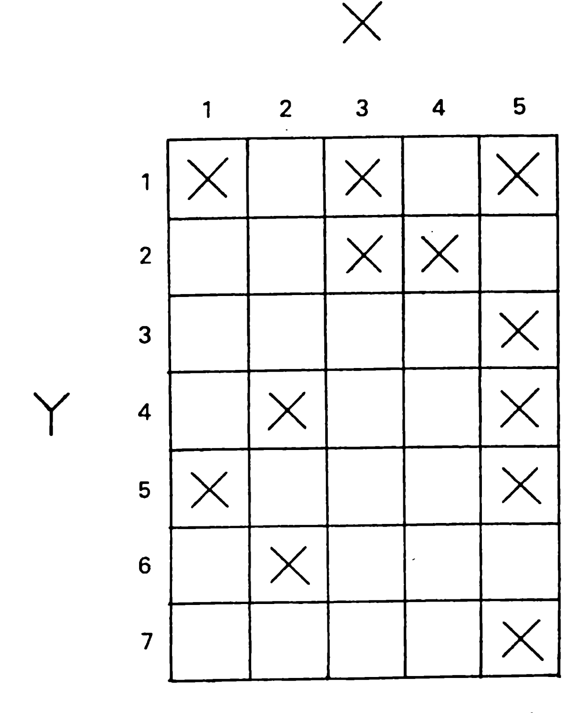


Fig. 8.3. Puntos utilizados en el árbol de decisión

```
  10 GO SUB 9000
9000 CLS
9010 FOR X=1 TO 7
9020 FOR Y=1 TO 9
9030 PRINT AT Y,X; PAPER 0;" ";
9040 NEXT Y: NEXT X
9050 FOR X=2 TO 6
9060 FOR Y=2 TO 8
9070 PRINT AT Y,X;; PAPER 6;" ";
9080 NEXT Y: NEXT X
9090 LET X=2: LET Y=2
9200 RETURN
```

Ahora se produce un cursor intermitente para mostrar la posición en la que nos hallamos. Se encuentra el carácter que hay en tal posición, SCREEN$(Y,X), y se almacena en la variable alfanumérica

C$. En esta posición se pone un signo de número (#) y a continuación se repone el carácter original C$ para que no exista un efecto duradero.

```
  20 LET A$=INKEY$
  30 LET C$=SCREEN$ (Y,X): PRINT
 AT Y,X;"#": PAUSE 10: PRINT AT
Y,X;C$
  40 IF A$="" THEN  GO TO 20
```

Las coordenadas X e Y se actualizan de acuerdo con el movimiento de las teclas del cursor (w, q, l, p) y cuando se pulsa la barra espaciadora se pone un asterisco en tal posición. Si se comete un error, pulsando la «x» se borra la posición actual, imprimiéndose un espacio vacío, y con la «Z» mayúscula ('z' + CAPS/SHIFT) se salta a la rutina de presentación de la cuadrícula y se borra la existente. Pulsando ENTER (CHR$(13)) se va a la rutina decodificadora, y en otro caso el programa retorna a la comprobación de teclado.

```
  50 IF A$="w" THEN  LET X=X+1
  60 IF A$="q" THEN  LET X=X-1
  70 IF A$="l" THEN  LET Y=Y+1
  80 IF A$="p" THEN  LET Y=Y-1
  90 IF A$=" " THEN  PRINT AT Y,
X;"*";
 100 IF A$="x" THEN  PRINT AT Y,
X;" ";
 110 IF A$="Z" THEN  GO SUB 9010
 120 IF A$=CHR$ (13) THEN  GO TO
 2000
 170 GO TO 20
```

Hay que establecer límites para impedir que el cursor se salga de la zona de la cuadrícula de 5 × 7.

```
130 IF X<2 THEN  LET X=2
140 IF X>6 THEN  LET X=6
150 IF Y<2 THEN  LET Y=2
160 IF Y>8 THEN  LET Y=8
```

El mantenimiento del árbol de decisión

El árbol de decisión se mantiene en una serie de matrices enlazadas en las que NB es el número de ramas. Los nombres de las letras se tienen en la variable L$(N). J(N) contiene la abscisa X que se ha de analizar a continuación y K(N) la correspondiente ordenada. El siguiente elemento a considerar, si la respuesta es «no», se guarda en L(N), y en M(N) el correspondiente para el caso de respuesta afirmativa.

```
9100 LET NB=53
9110 RESTORE : DIM L$(NB,1): DIM
 J(NB): DIM K(NB): DIM L(NB): DI
M M(NB)
9120 FOR N=1 TO NB
9130 READ L$(N),J(N),K(N),L(N),M
(N)
9140 NEXT N
```

La mejor forma de aplicar los datos, con la instrucción DATA, es por medio de 53 líneas separadas (una para cada punto de la rama) ya que esto facilita la entrada y pone en evidencia los errores. Desgraciadamente el BASIC Sinclair requiere que hayan de entrarse las cadenas vacías o ceros donde no se desea entrar ningún valor.

```
9300 DATA "",1,1,2,19
9310 DATA "",1,5,3,10
9320 DATA "",3,2,4,9
9330 DATA "",5,1,5,8
9340 DATA "",3,1,6,7
9350 DATA " ",0,0,0,0
9360 DATA "S",0,0,0,0
9370 DATA "J",0,0,0,0
9380 DATA "I",0,0,0,0
9390 DATA "",5,4,11,9
9400 DATA "",5,5,12,13
9410 DATA "C",0,0,0,0
9420 DATA "G",0,0,0,0
```

```
9430 DATA "",5,7,18,15
9440 DATA "",2,4,17,16
9450 DATA "A",0,0,0,0
9460 DATA "Q",0,0,0,0
9470 DATA "O",0,0,0,0
9480 DATA "",5,1,20,29
9490 DATA "",5,4,21,28
9500 DATA "",5,3,27,22
9510 DATA "",5,7,23,26
9520 DATA "",5,5,24,25
9530 DATA "P",0,0,0,0
9540 DATA "B",0,0,0,0
9550 DATA "R",0,0,0,0
9560 DATA "L",0,0,0,0
9570 DATA "D",0,0,0,0
9580 DATA "",5,7,45,30
9590 DATA "",2,6,31,44
9600 DATA "",5,3,32,39
9610 DATA "",1,5,33,36
9620 DATA "",3,1,34,35
9630 DATA "X",0,0,0,0
9640 DATA "Z",0,0,0,0
9650 DATA "",4,2,38,37
9660 DATA "K",0,0,0,0
9670 DATA "E",0,0,0,0
9680 DATA "",2,4,40,43
9690 DATA "",4,2,42,41
9700 DATA "M",0,0,0,0
9710 DATA "N",0,0,0,0
9720 DATA "H",0,0,0,0
9730 DATA "W",0,0,0,0
9740 DATA "",3,1,46,51
9750 DATA "",1,5,47,50
9760 DATA "",2,4,48,49
9770 DATA "Y",0,0,0,0
9780 DATA "V",0,0,0,0
9790 DATA "U",0,0,0,0
9800 DATA "",1,5,52,53
9810 DATA "T",0,0,0,0
9820 DATA "F",0,0,0,0
```

Si se tiene confianza en uno mismo (o se trata de ahorrar espacio) pueden condensarse los datos en un número más pequeño de líneas de difícil lectura pero es muy fácil que se cometan errores.

La comprobación del gráfico realizado

Para comprobar el dibujo producido comparándolo con los modelos disponibles (véase el Diagrama de flujo 8.1) se pone en 1 el puntero AP con objeto de que el análisis se empiece desde el principio. Se leen las coordenadas X e Y de los elementos de J(AP) y K(AP) señalados por el puntero, y la última posición LP se hace igual al valor del puntero AP de la matriz presente.

Se determina ahora el carácter en esas coordenadas mediante SCREEN$(X,Y). Si se trata de un asterisco, es que el punto se ha marcado y debe seguir el puntero afirmativo (de «síes»), M(AP). Si se encuentra cualquier otro valor, sigue el puntero de los «noes», L(AP). En cualquier caso se hace una comprobación para ver si el elemento señalado contiene un cero (que indica el final de una rama), lo que expresa que se ha encontrado el carácter que se buscaba. En este caso, se presenta en pantalla la letra o carácter en cuestión —L$(LP)— y se mantiene la presentación hasta que se pulse una tecla, con lo que se inicia un nuevo ciclo. Si aparece un valor superior a cero debe tratarse de otro punto de la rama por lo que el programa salta y recoge los nuevos valores de J(AP) y K(AP).

Para que se puedan ver los puntos que se han comprobado se pone el signo «@» a medida que se van encontrando. Cualquier punto establecido pero no comprobado permanecerá con un asterisco.

```
2000 LET AP=1
2010 LET X=J(AP)+1: LET Y=K(AP)+
1: LET LP=AP
2020 LET P$=SCREEN$ (Y,X)
2030 IF P$="*" THEN  LET AP=M(AP
): GO TO 2050
2040 LET AP=L(AP)
2050 IF AP=0 THEN  GO TO 2070
```

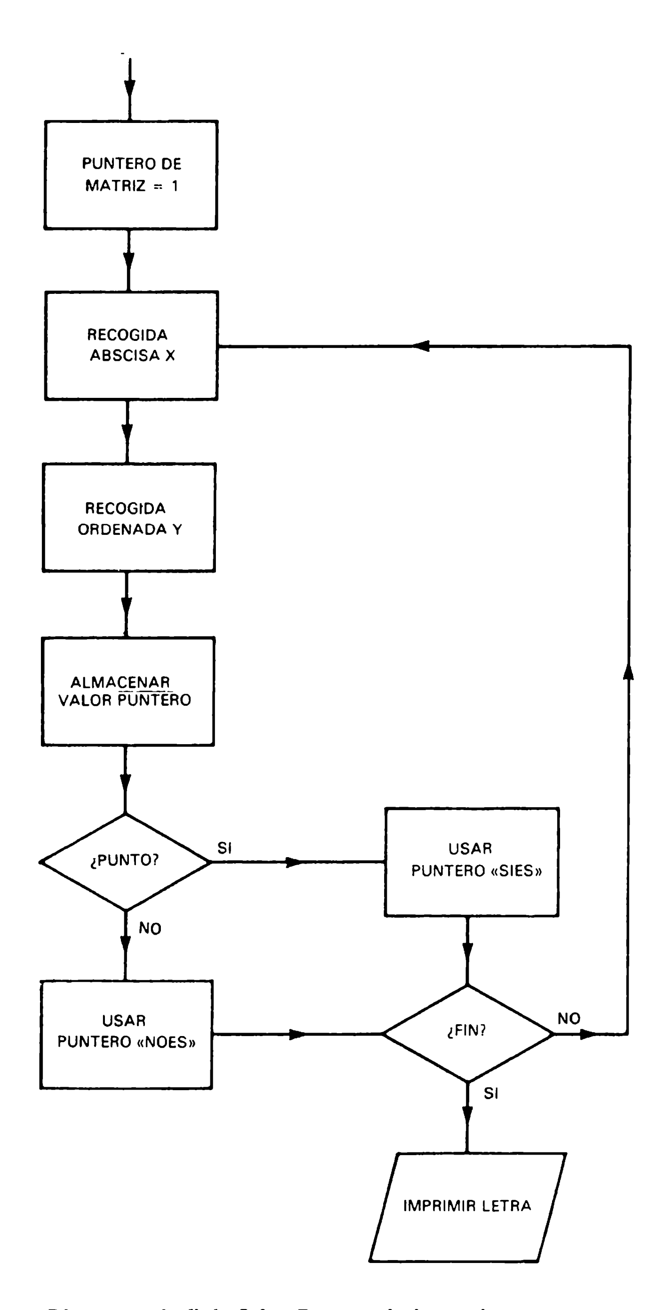


Diagrama de flujo 8.1. Reconocimiento de caracteres

```
2060 PRINT AT Y,X;"@";: GO TO 20
10
2070 PRINT AT 12,4;L$(LP);
2080 PAUSE 0
2090 GO SUB 9000: GO TO 20
```

Si se desea saber qué parte del árbol de decisión se ha seguido realmente, añádanse estas modificaciones que presentarán la secuencia seguida.

```
2000 LET AP=1: LET A=3: PRINT AT
 1,15;"AP"
2050 PRINT AT A,15;AP: LET A=A+1
: IF AP=0 THEN  GO TO 2070
```

El inconveniente de este método rápido, que sólo analiza ciertos puntos críticos, es que hará una adaptación equivocada si encuentra una forma que no se halla en el árbol, mientras que si se comprueban todos los puntos no ocurrirá tal cosa.

Los primeros Lectores Opticos de Caracteres sólo aceptaban un tipo de letra pero los últimos modelos no sólo admiten tipos de diferentes estilos, sino que aprenden por sí mismos las reglas para el reconocimiento al contar con un sistema experto incorporado. La forma de enseñarles es mostrarles algunas páginas de texto y, a continuación, se les introducen los mismos caracteres a través del teclado. Creemos, sin embargo, que aún pasará bastante tiempo antes de que cualquiera pueda crear una máquina que entienda nuestra letra manuscrita.

Sigue a continuación el listado completo para el reconocimiento de formas gráficas:

```
  10 GO SUB 9000
  20 LET A$=INKEY$
  30 LET C$=SCREEN$ (Y,X): PRINT
 AT Y,X;"#": PAUSE 10: PRINT AT
Y,X;C$
  40 IF A$="" THEN  GO TO 20
  50 IF A$="w" THEN  LET X=X+1
```

```
  60 IF A$="q" THEN  LET X=X-1
  70 IF A$="l" THEN  LET Y=Y+1
  80 IF A$="p" THEN  LET Y=Y-1
  90 IF A$=" " THEN  PRINT AT Y,
X;"*";
 100 IF A$="x" THEN  PRINT AT Y,
X;" ";
 110 IF A$="Z" THEN  GO SUB 9010
 120 IF A$=CHR$ (13) THEN  GO TO
 2000
 130 IF X<2 THEN  LET X=2
 140 IF X>6 THEN  LET X=6
 150 IF Y<2 THEN  LET Y=2
 160 IF Y>8 THEN  LET Y=8
 170 GO TO 20
2000 LET AP=1: LET A=3: PRINT AT
 1,11;"PUNTERO DE"
2005 PRINT AT 2,10;"MATRICES"
2010 LET X=J(AP)+1: LET Y=K(AP)+
1: LET LP=AP
2020 LET P$=SCREEN$ (Y,X)
2030 IF P$="*" THEN  LET AP=M(AP
): GO TO 2050
2040 LET AP=L(AP)
2050 PRINT AT A,15;AP: LET A=A+1
: IF AP=0 THEN  GO TO 2070
2060 PRINT AT Y,X;"@";: GO TO 20
10
2070 PRINT AT 12,4;L$(LP);
2080 PAUSE 0
2090 GO SUB 9000: GO TO 20
9000 CLS
9010 FOR X=1 TO 7
9020 FOR Y=1 TO 9
9030 PRINT AT Y,X; PAPER 0;" ";
9040 NEXT Y: NEXT X
9050 FOR X=2 TO 6
9060 FOR Y=2 TO 8
9070 PRINT AT Y,X;; PAPER 6;" ";
9080 NEXT Y: NEXT X
```

```
9090 LET X=2: LET Y=2
9100 LET NB=53
9110 RESTORE : DIM L$(NB,1): DIM
 J(NB): DIM K(NB): DIM L(NB): DI
M M(NB)
9120 FOR N=1 TO NB
9130 READ L$(N),J(N),K(N),L(N),M
(N)
9140 NEXT N
9200 RETURN
9300 DATA "",1,1,2,19
9310 DATA "",1,5,3,10
9320 DATA "",3,2,4,9
9330 DATA "",5,1,5,8
9340 DATA "",3,1,6,7
9350 DATA " ",0,0,0,0
9360 DATA "S",0,0,0,0
9370 DATA "J",0,0,0,0
9380 DATA "I",0,0,0,0
9390 DATA "",5,4,11,9
9400 DATA "",5,5,12,13
9410 DATA "C",0,0,0,0
9420 DATA "G",0,0,0,0
9430 DATA "",5,7,18,15
9440 DATA "",2,4,17,16
9450 DATA "A",0,0,0,0
9460 DATA "Q",0,0,0,0
9470 DATA "O",0,0,0,0
9480 DATA "",5,1,20,29
9490 DATA "",5,4,21,28
9500 DATA "",5,3,27,22
9510 DATA "",5,7,23,26
9520 DATA "",5,5,24,25
9530 DATA "P",0,0,0,0
9540 DATA "B",0,0,0,0
9550 DATA "R",0,0,0,0
9560 DATA "L",0,0,0,0
9570 DATA "D",0,0,0,0
9580 DATA "",5,7,45,30
9590 DATA "",2,6,31,44
9600 DATA "",5,3,32,39
```

```
9610 DATA "",1,5,33,36
9620 DATA "",3,1,34,35
9630 DATA "X",0,0,0,0
9640 DATA "Z",0,0,0,0
9650 DATA "",4,2,38,37
9660 DATA "K",0,0,0,0
9670 DATA "E",0,0,0,0
9680 DATA "",2,4,40,43
9690 DATA "",4,2,42,41
9700 DATA "M",0,0,0,0
9710 DATA "N",0,0,0,0
9720 DATA "H",0,0,0,0
9730 DATA "W",0,0,0,0
9740 DATA "",3,1,46,51
9750 DATA "",1,5,47,50
9760 DATA "",2,4,48,49
9770 DATA "Y",0,0,0,0
9780 DATA "V",0,0,0,0
9790 DATA "U",0,0,0,0
9800 DATA "",1,5,52,53
9810 DATA "T",0,0,0,0
9820 DATA "F",0,0,0,0
```

Información para la ejecución del Programa de Identificación de Formas Gráficas

Una vez establecida la letra mayúscula que vamos a identificar, tenemos que determinar los puntos de la cuadrícula que han de llevar un asterisco. Los buscaremos en el Arbol de Decisión desde el lugar en el que se encuentra la letra, descendiendo hasta el tronco inicial, el 1. Anotaremos el número de los sucesivos entronques con las correspondientes coordenadas y si al nudo se llega a través de un SI o de un NO. Los correspondientes al SI nos dan las coordenadas de la cuadrícula de la pantalla donde se pondrá el asterisco.

Ejemplo:

Letra seleccionada: S
Entronque 7

Entronques descendentes	Ramas NO	Ramas SI	Coordenadas X	Coordenadas Y
7				
5		SI	3	1
4	NO			
3	NO			
2	NO			
1	NO			

Llevaremos el cursor accionando las letras correspondientes (w → un salto a la derecha; q → un salto a la izquierda; p → un salto arriba; l → un salto abajo) a las coordenadas 3,1 y se marcará allí el asterisco, que para este caso basta con uno. En pantalla nos aparecerán señalados los lugares investigados que son los nudos determinados.

9

La Inteligencia Artificial en la enseñanza

Otro campo en el que la aplicación de la Inteligencia Artificial es especialmente útil es en el de los programas de enseñanza. Resulta muy interesante disponer de un programa para determinar el nivel de conocimientos de un alumno pero la actuación de un profesor humano no se limita a esto. El profesor no hace sólo preguntas sino que controla los adelantos del alumno, aumenta progresivamente la dificultad de las preguntas, a medida que se va avanzando, y somete al alumno a pruebas más rigurosas sobre los tipos de problemas en los que se encuentra más dificultades. Por ejemplo, si un alumno es examinado sobre problemas de adición, sustracción, multiplicación y división, pero sólo se equivoca en la última de tales operaciones, debe deducirse que necesita hacer divisiones para adquirir más práctica con las mismas.

Preguntas y respuestas

Necesitamos crear números aleatorios para ser utilizados en una suma. Con la instrucción INT(RND*10) obtendremos números del 0 al 9.

```
20 LET A=INT (RND*10)
30 LET B=INT (RND*10)
```

El ordenador los suma y después va a la subrutina 1000 para la comprobación.

```
40 LET C=A+B: GO SUB 1000
```

La subrutina primero debe presentar la pregunta y recibir mediante la instrucción INPUT la respuesta (IP).

```
1000 PRINT A;"+";B;"=";
1010 INPUT IP: PRINT IP
```

La respuesta debe comprobarse y si C es igual a ella se presentará la palabra CORRECTO y la rutina regresa a la línea 40. En otro caso, se dirá EQUIVOCADO, seguido del valor corregido.

```
1020 IF C=IP THEN  PRINT "CORREC
TO": RETURN
1030 PRINT "EQUIVOCADO, LA RESPU
ESTA CORRECTA ES ";C
1040 RETURN
```

Los temas de sustracción, multiplicación y división se desarrollan de manera análoga si se substituye el signo « + » de la línea 1000 por el símbolo de cadena —S$— al que se puede aplicar el carácter apropiado al caso. Como la instrucción INT(RND*10) es común para todos los cálculos, pudiera definirse como una función (RD).

```
  10 LET X=0
  15 DEF FN R(X)=INT (RND*10)
  20 LET A=FN R(X)
  30 LET B=FN R(X)
  40 LET S$="+": LET C=A+B: GO S
UB 1000
  50 LET A=FN R(X)
  60 LET B=FN R(X)
  70 LET S$="-": LET C=A-B: GO S
UB 1000
```

```
  80 LET A=FN R(X)
  90 LET B=FN R(X)
 100 LET S$="*": LET C=A*B: GO S
UB 1000
 110 LET A=FN R(X)
 120 LET B=FN R(X)
 130 LET S$="/": LET C=A/B: GO S
UB 1000
1000 PRINT A;S$;B;"=";
```

Finalmente saltamos a la línea 20 para pedir más problemas.

```
140 GO TO 20
```

La división por cero

Tal como está el programa puede interrumpirse si B es un cero en el caso de una división por cero. Esto puede arreglarse añadiendo simplemente uno a B en este caso.

```
120 LET B=FN R(X)+1
```

El borrado de decimales

Estamos utilizando variables enteras para trabajar con esta clase de números pero una división puede generar decimales que no pueden entrarse correctamente ya que IP se redondeará.

3/2 = 1,5

Pero el programa aceptará 1, 1,5, 1,9 ó cualquier otro valor entre 1 y 1,999... como correctos.

Para evitar que se obtengan decimales A ha de ser múltiplo de B. Para hacerlo, calculamos primero B y hacemos que A sea igual a B multiplicado por un número aleatorio entre 0 y 10.

```
110 LET B=FN R(X)+1
120 LET A=INT (FN R(X))*B
```

La utilización de un sistema de puntuación o tanteo

Ahora que ya tenemos la prueba preparada, hemos de considerar la introducción de un sistema de puntuación o tanteo. La forma más sencilla es incrementar una variable TR cada vez que se utiliza la subrutina de la línea 1000 y hacer lo mismo con una variable de tanteo SC cada vez que se logra una respuesta correcta.

```
  10 LET X=0: LET TR=0: LET SC=0
1010 INPUT IP: LET TR=TR+1
1020 IF C=IP THEN  PRINT "CORREC
TO ": LET SC=SC+1: GO TO 1040
1040 PRINT "TU PUNTUACION ES ";S
C;"/";TR: RETURN
```

Si se prefiere el tanteo en forma de porcentaje, se enmienda la línea 1040 como sigue:

```
1040 PRINT "HAS TENIDO UN ";INT
((SC/TR)*100);" POR CIENTO DE  A
CIERTOS ": RETURN
```

La posibilidad de crear un número infinito de preguntas

Tal como está, el programa, puede hacer una pregunta de cada tipo, en forma secuencial, hasta el infinito. Podemos limitar el número de preguntas definiéndolo como una variable NQ.

```
  10 LET X=0: LET TR=0: LET SC=0
: LET NQ=32
```

Cada vez que se hace una pregunta, se decrementa en uno NQ y cuando NQ = 0, termina la prueba (después de haberse respondido a ocho preguntas de cada tipo).

```
 140 IF NQ>0 THEN  GO TO 20
 160 STOP
1010 INPUT IP: LET TR=TR+1: LET
NQ=NQ-1
```

La valoración de las áreas de dificultad

Si hemos de considerar cada tema en función de áreas de dificultad hay que llevar un registro de los resultados logrados en cada una de ellas. Necesitamos pues variables separadas para cada tipo de operaciones (AD para la suma, SU para la sustracción, MU para la multiplicación y DI para la división). Estas variables se definen en términos de un octavo del número total de problemas propuestos: NQ.

```
  10 LET X=0: LET TR=0: LET SC=0
: LET NQ=32: LET AD=NQ/8: LET SU
=AD: LET MU=AD: LET DI=AD
```

Ahora si la respuesta correcta C es la misma que la entrada —IP—, una variable de incremento IN es puesta a —1, se presenta en pantalla CORRECTO y regresa la rutina. En caso negativo, IN se pone en 1 y en pantalla aparece EQUIVOCADO seguido del valor correcto.

```
1020 IF C=IP THEN  LET IN=-1: PR
INT "CORRECTO": RETURN
1030 LET IN=1: PRINT "ERROR, LA
RESPUESTA CORRECTA ES ";C
1040 RETURN
```

A la variable que corresponda de los temas AD, SU, MU o DI, se incrementa IN haciendo que su valor crezca si la respuesta es incorrecta o disminuya cuando es correcta.

```
  40 LET S$="+": LET C=A+B: GO S
UB 1000: LET AD=AD+IN
  70 LET S$="-": LET C=A-B: GO S
UB 1000: LET SU=SU+IN
 100 LET S$="*": LET C=A*B: GO S
UB 1000: LET MU=MU+IN
 130 LET S$="/": LET C=A/B: GO S
UB 1000: LET DI=DI+IN
```

Añadimos ahora una comprobación para ver si todas las preguntas de un determinado tipo no han sido respondidas correctamente (es decir, AD>0, véase el Diagrama de Flujo 9.1). Si lo han sido, ya no se proponen más de este tipo al saltarse la línea. Si el número apropiado de cada clase ha sido correctamente respondido (AD = 0, SU = 0, MU = 0, DI = 0) el programa termina.

```
  40 IF AD>0 THEN  LET S$="+": L
ET C=A+B: GO SUB 1000: LET AD=AD
+IN
  70 IF SU>0 THEN  LET S$="-": L
ET C=A-B: GO SUB 1000: LET SU=SU
+IN
 100 IF MU>0 THEN  LET S$="*": L
ET C=A*B: GO SUB 1000: LET MU=MU
+IN
 130 IF DI>0 THEN  LET S$="/": L
ET C=A/B: GO SUB 1000: LET DI=DI
+IN
 140 IF AD=0 AND SU=0 AND MU=0 A
ND DI=0 THEN  GO TO 160
```

Obsérvese que ya no se hacen preguntas sobre temas en los que se haya respondido adecuadamente a cuatro problemas. Si se da un error, entonces AD, por ejemplo, se incrementa y, por consiguiente, se tendrán que responder más de cuatro antes de que AD se haga cero.

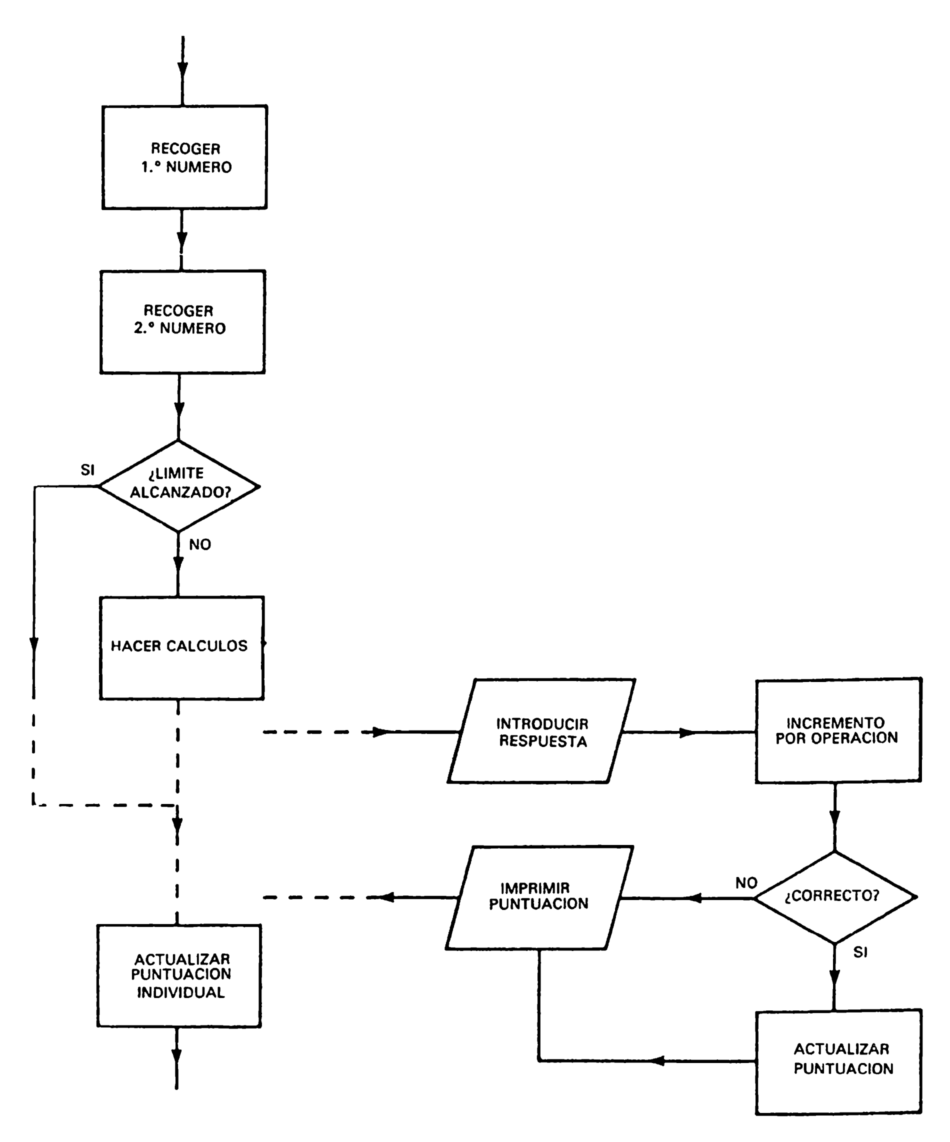


Diagrama de flujo 9.1. El Profesor Inteligente

El establecimiento de niveles de dificultad

¿Cómo hacer que las preguntas sean más o menos difíciles según se vayan respondiendo? Hasta ahora los valores de A y B han estado

comprendidos entre 0 y 9 ya que se derivaban de la instrucción INT (RND*10), pero ahora debemos tender hacia valores más altos si se logran respuestas correctas o más bajos, en caso contrario. Al mismo tiempo, tenemos que asegurarnos que no se produzcan valores negativos si se incide en el error.

El caso peor sería si se respondieran mal todas las preguntas del último grupo. En este caso sólo se plantearían cuatro preguntas de los tres primeros grupos, dejando 32 — (3*4) = 20 para el último. Además debemos recordar que X (AD, por ejemplo) empieza con el valor 4 por lo que el máximo de X que se podría obtener sería 20 + 4 = 24.

Establecemos, por consiguiente, una variable de ponderación WT que se calcula sustrayendo tres veces el número de preguntas de cada grupo (3*AD) del total de preguntas (NQ) y volviendo a añadir el número de preguntas del grupo AD:

WT = NQ — (3*AD) + AD

Esto se expresa más sencillamente así:

WT = NQ — (2*AD)

```
  10 LET X=0: LET TR=0: LET NQ=3
2: LET SC=0: LET AD=NQ/8: LET SU
=AD: LET MU=AD: LET DI=AD: LET W
T=NQ-(2*AD)
  20 LET A=FN R(AD)
  30 LET B=FN R(AD)
  50 LET A=FN R(SU)
  60 LET B=FN R(SU)
  80 LET A=FN R(MU)
  90 LET B=FN R(MU)
 110 LET B=FN R(DI)+1
 120 LET A=INT (FN R(DI))*B
```

Ahora substituimos el valor fijo de 10 por la diferencia entre WT y X.

```
 15 DEF FN R(X)=INT (RND*(WT-X))
```

Para empezar, WT = 24 y X = 4 por lo que se seleccionarán los números entre 0 y 19. Si se da una respuesta correcta, se reduce X a 3 y se seleccionan valores entre 0 y 20. Después de cuatro respuestas correctas, X ya no cambiará para este tipo de preguntas ya que habrá llegado a cero y la línea será soslayada. Los últimos valores serán, por consiguiente, entre 0 y 22. Sin embargo, si la primera respuesta es incorrecta, X se incrementa en 1 y la escala de números producidos se reduce igualmente en 1 (0 → 18). En el caso peor, X aumentará 20 veces hasta 24 y (WT — X) caerá a cero para A y B (¡Usted debe resolver este problema particular!).

A continuación sigue el listado completo del programa para facilitar la composición de los fragmentos que se han ido presentando a lo largo del capítulo. Se ha incorporado la línea 1015 para poder salir fácilmente del programa si se desea.

```
  10 LET X=0: LET TR=0: LET NQ=3
2: LET SC=0: LET AD=NQ/8: LET SU
=AD: LET MU=AD: LET DI=AD: LET W
T=NQ-(2*AD)
  15 DEF FN R(X)=INT (RND*(WT-X)
)
  20 LET A=FN R(AD)
  30 LET B=FN R(AD)
  40 IF AD>0 THEN  LET S$="+": L
ET C=A+B: GO SUB 1000: LET AD=AD
+IN
  50 LET A=FN R(SU)
  60 LET B=FN R(SU)
  70 IF SU>0 THEN  LET S$="-": L
ET C=A-B: GO SUB 1000: LET SU=SU
+IN
  80 LET A=FN R(MU)
  90 LET B=FN R(MU)
 100 IF MU>0 THEN  LET S$="*": L
ET C=A*B: GO SUB 1000: LET MU=MU
+IN
 110 LET B=FN R(DI)+1
 120 LET A=INT (FN R(DI))*B
 130 IF DI>0 THEN  LET S$="/": L
```

```
ET C=A/B: GO SUB 1000: LET DI=DI
+IN
 140 IF AD=0 AND SU=0 AND MU=0 A
ND DI=0 THEN  GO TO 160
 150 GO TO 15
 160 STOP
1000 PRINT 'A;S$;B;"=";
1010 INPUT "PON TU RESPUESTA: (0
.5 SI        DESEAS TERMINAR): ";
IP: LET TR=TR+1: LET NQ=NQ-1
1015 IF IP=.5 THEN  STOP
1020 IF C=IP THEN  LET IN=-1: PR
INT ;C;";";" CORRECTO": LET SC=S
C+1: GO TO 1040
1030 LET IN=1: PRINT IP;";";" ER
ROR, LA RESPUESTA             CO
RRECTA ES ";C
1040 PRINT '"TU PUNTUACION ES DE
 ";SC;" PUNTOS,   QUE CORRESPOND
E A UN PORCENTAJE  DE ACIERTOS D
EL ";INT ((SC/TR)*100);" POR CIE
NTO.": RETURN
```

10

La realización de un programa completo de Inteligencia Artificial

En los capítulos anteriores hemos tratado, desde los primeros principios, los diversos aspectos de la Inteligencia Artificial. En este último vamos a relacionar entre sí muchas de las ideas de aquéllos en un solo programa completo.

El programa «inteligente» original fue el famoso «ELIZA» que se escribió para establecer un determinado estilo de terapia psiquiátrica. Nos hemos resistido a la tentación de seguir aquel ejemplo y hemos optado, en su lugar, por producir la sustitución por un típico vendedor de ordenadores. Este programa combina algunas ideas sobre el proceso del lenguaje natural y de los sistemas expertos para llegar a un resultado que permite comprender las peticiones y hacer sugerencias que tienen en cuenta los requisitos solicitados y algunas realidades comerciales.

La posibilidad de adaptar el programa a necesidades particulares

Se ha incluido un número suficiente de palabras y valores para hacer el programa interesante pero puede adaptarse al gusto de cada uno añadiendo las ideas propias. (No nos hacemos responsables de los valores que se incluyen —que únicamente tienen fines demostrativos— ni de los criterios sobre determinadas máquinas expresados en el programa que, por otra parte, se ponen con nombres simulados). El

programa, en realidad, resulta bastante complicado pero sigue los métodos descritos anteriormente en el libro. En la Tabla 10.1 se dan las variables principales.

Tal como está el programa, cabe justamente en un Spectrum de 16 K. (Si se dispone de más memoria no hay problemas y puede ampliarse el programa si se desea). Sin embargo, ya que se consume hasta el último byte de memoria en la máquina de 16 K, puede ocurrir que el programa se detenga con el mensaje «4 Out of memory» («memoria consumida») durante la comparación de las cadenas si las frases de entrada son largas. No es posible introducir un litro en un recipiente de medio, por lo que si se desea aplicar frases muy largas habrá que cortar por alguna parte (puede hacerse, por ejemplo, reduciendo el tamaño de las matrices poniendo menos características; limítese el valor de FE en la línea 9999 a 16 y suprímanse los dos últimos elementos de las instrucciones DATA de las líneas 940 y 1010).

(Nota del T.: La versión traducida ha sido ejecutada y no requiere tanta memoria como da a entender el autor).

Tabla 10.1. Principales Variables del Programa

N$	cadena de entrada
WL	longitud de palabra
J$	cadena de análisis
I$	cadena del objeto a buscar (recogida de N$)
SP	puntero de búsqueda
FT	principio de la búsqueda
PH	número de la frase
QP	número de las diferentes preguntas preparadas
R$	nombre de la característica analizada
Q$	petición actual
Q	número de orden de la característica analizada
CR	tasa de coste analizada
PR	tasa de beneficio analizada
TP	beneficio total
TC	coste total
R	número de reglas

RU	marcador de la regla de actualización
OB	número de objetos o características que se analizan
AJ	número de adjetivos
AV	número de adverbios
LI	número de verbos favorables
DL	número de verbos desfavorables
NJ	número de adjetivos negativos
NV	número de adverbios negativos
HM	número de adjetivos de barato y caro
BB	saldo bancario
FE	número de características
CT	número de precios de coste
CS	número de sugerencias sobre el coste
EX	número de excusas
HI	número de sugerencias de precio alto
LO	número de sugerencias de precio bajo
LD	valor de los términos favorables / desfavorables
NP	factores negativos
OF	señalizador del objeto
OM	orden del objeto identificado
D$	lista de costes de los ordenadores
R(N)	regla de decisión
I(N)	regla de beneficio
J(N)	regla de costes

El diálogo usuario-programa

El método que sigue el programa es el de preguntar la opinión sobre cada posible característica una por una. La forma concreta de hacer la pregunta se escoge aleatoriamente entre una selección de frases. Téngase en cuenta que la palabra clave o la frase fundamental se inserta en la pregunta donde es necesario a plena concordancia gramatical (o casi plena).

La respuesta entrada se examina detalladamente en busca de las palabras claves y la matriz de control se actualiza según los requi-

sitos pedidos. Un BEEP corto suena cada vez que se acude a la rutina de identificación (INSTR) (Línea 10) para que se sepa que el ordenador no se ha dormido cuando analiza una frase larga. Se presenta en pantalla la matriz de control para que pueda verse cómo se actualiza. La primera fila de valores es la regla de decisión, la segunda la de costes y la tercera la de beneficios. Muchas de las palabras claves se han truncado para que puedan igualmente identificarse los derivados de las mismas.

La división de las palabras en clases diferentes

La respuesta más sencilla es «SI» o «NO», que suma o resta una unidad al valor de la regla de la característica en cuestión. Si se menciona su nombre (GRAFICOS, por ejemplo) se añade otro 1 a la regla. Usando, además, un adjetivo o un adverbio positivo (favorable), también se incrementa, mientras que si son negativos (desfavorables), se produce una reducción. Separando las palabras en distintas clases se permite hacer más de un cambio de la regla al mismo tiempo.

Así:

SI	suma uno
SI GRAFICOS	suma dos
SI SONIDO NECESARIO	suma tres
SI BUEN TECLADO NECESARIO	suma cuatro

Mientras que:

NO	resta uno
NO NECESITO MEMORIA	resta dos

La clasificación de los verbos

Además, los verbos están agrupados en positivos (favorables) y negativos (desfavorables). Estos invierten la acción del resto de las palabras.

Así:

YO DETESTO LOS MICRODRIVES	resta dos
NO ME GUSTA EL SONIDO	resta dos
YO NO ODIO EL SONIDO	suma dos

Si aparece algo al principio de una frase, separado por una coma, se ignora lo que está delante de ella.

Ejemplo:

NO, SONIDO	suma dos a la regla del sonido

La función de la coma

La excepción es cuando se incluyen «Y» o «PERO» en la oración con coma pues ambas partes actúan independientemente.

Así, si la pregunta es:

DESEA USTED UNOS BUENOS GRAFICOS?

y la respuesta es:

NO, PERO SI UN BUEN BASIC

se resta uno de las reglas de GRAFICOS y se añaden dos a la del BASIC.

Si el programa no encuentra ninguna palabra clave en la expresión entrada, pedirá cortésmente que se repita nuevamente:

PERDON, PERO NO LE ENTIENDO

Análisis secuencial de las características

El programa sólo puede analizar una característica cada vez por lo que si se intenta, por ejemplo, en pedir SONIDO y GRAFICOS al mismo tiempo, el ordenador replicará:

POR FAVOR, CADA COSA A SU TIEMPO!

Sin embargo, es posible hacer comentarios sobre una sola característica distinta a la que se pregunta en esa ocasión y estas entradas actualizarán la regla correspondiente (como en el ejemplo anterior del PERO).

La toma de decisiones

Además de la matriz de control hay otras dos que están relacionadas con ella. La primera de éstas es la de costes que da una indicación del correspondiente a la opción analizada, y la segunda es la de beneficio que orienta al vendedor sobre el esfuerzo que vale la pena realizar para vender la característica en cuestión. Los valores de estas dos matrices se producen al multiplicar el contenido del elemento de la matriz de control por los factores introducidos originariamente como datos en la línea 1010, expresando, por este orden: frase que describe la característica, el coste y el beneficio.

Después de cada entrada, el vendedor considera las consecuencias de las peticiones que se le han hecho. En primer lugar, investiga si la suma total del coste de todos los requisitos solicitados excede el saldo bancario. En caso afirmativo, presenta en pantalla un mensaje seleccionado entre varios de carácter cáustico con respecto al valor del crédito. Por ejemplo:

CREO QUE SE ESTA SALIENDO DE SUS POSIBILIDADES

También analiza el beneficio que puede conseguir con esta venta tal como va por el momento. Si tal beneficio se reduce demasiado, empezará a perder interes y ofrecerá comentarios como éste:

TENGO UNA CITA URGENTE

O

CERRAMOS DENTRO DE CINCO MINUTOS

Al mismo tiempo, tratará de ayudar al comprador estableciendo cuál de los ordenadores se adapta a los requisitos solicitados. Para esto crea una lista en la que se comparan los valores aplicados en principio a la característica en cuestión en la descripción de cada ordenador, indicando el valor atribuido. El formato de las descripciones es:

(nombre, valor de la característica 1, valor de la característica 2, valor de la característica 3, etc.).

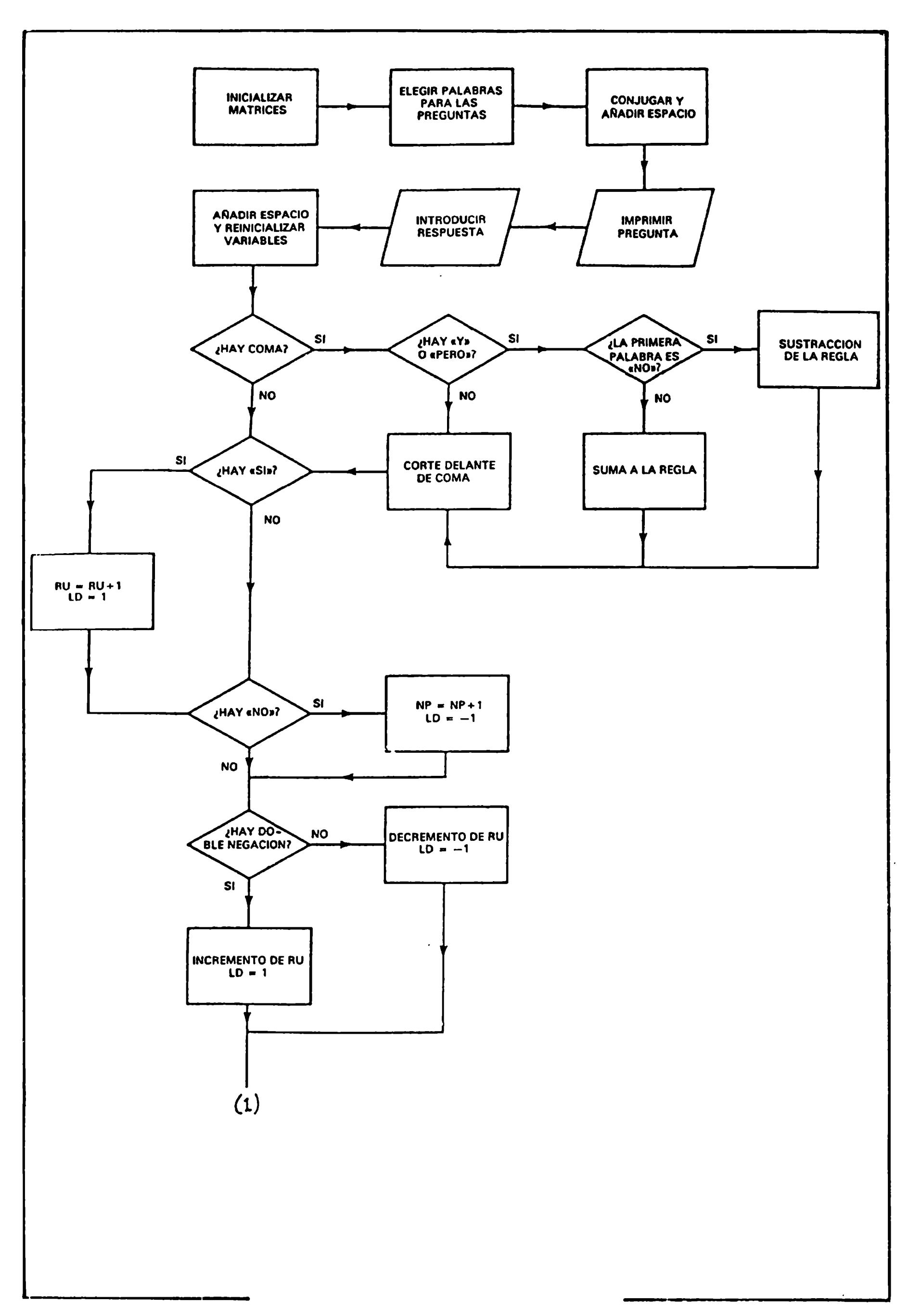
El ordenador mejor valorado se seleccionará siempre el primero pero, si es posible, se seleccionarán tres máquinas al menos (posiblemente con los promedios más bajos) y la elección final se efectuará de entre estas tres. Se seleccionará (aleatoriamente) para ser mencionado el más caro o el más barato de los ordenadores, por ejemplo:

SI DESEA UN LUJO CONSIDERE EL ORDENADOR...

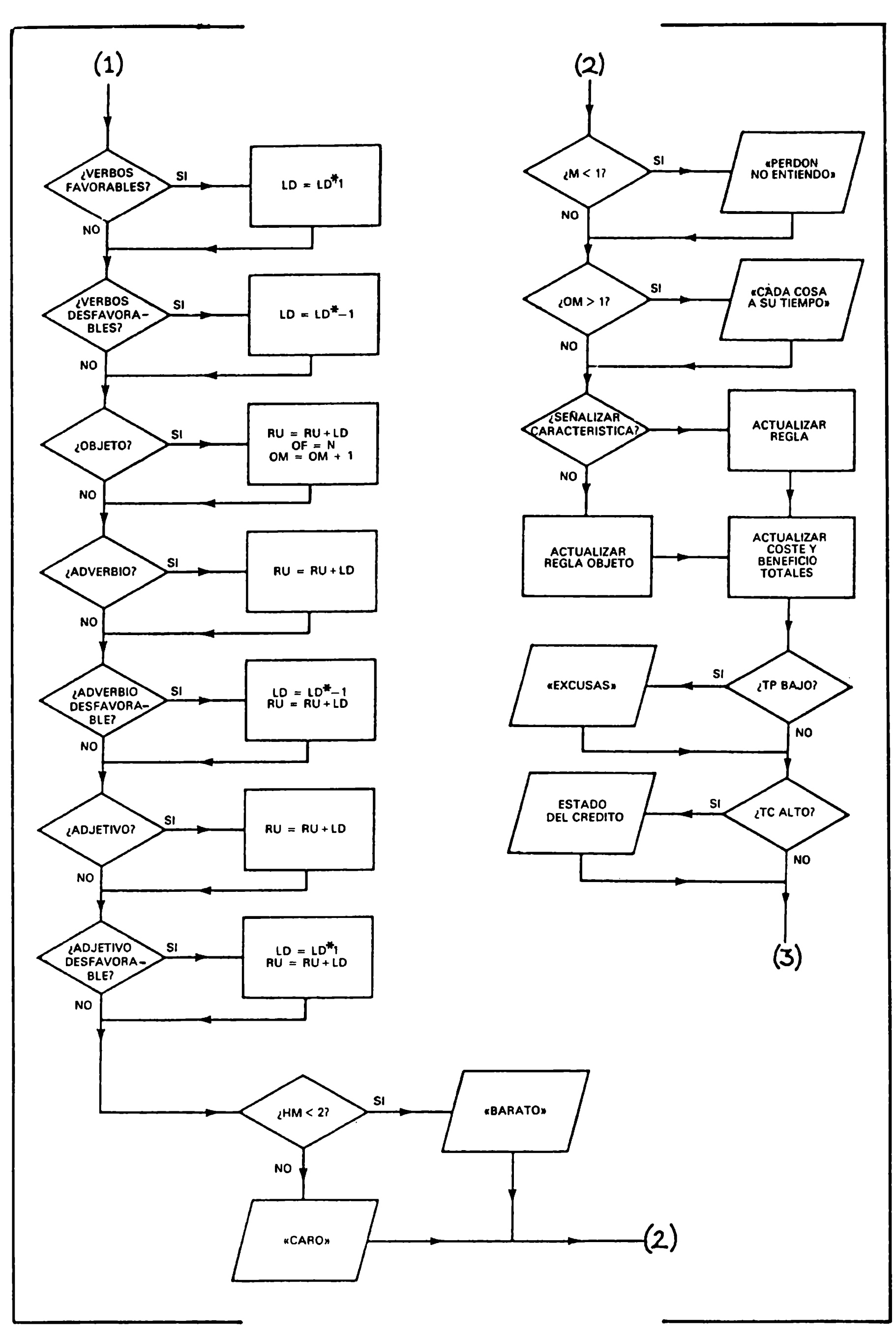
SI PUEDE PAGARLO QUE LE PARECE EL ORDENADOR...

Si sólo hay una máquina que se adapta a las posibilidades del comprador, el programa dirá:

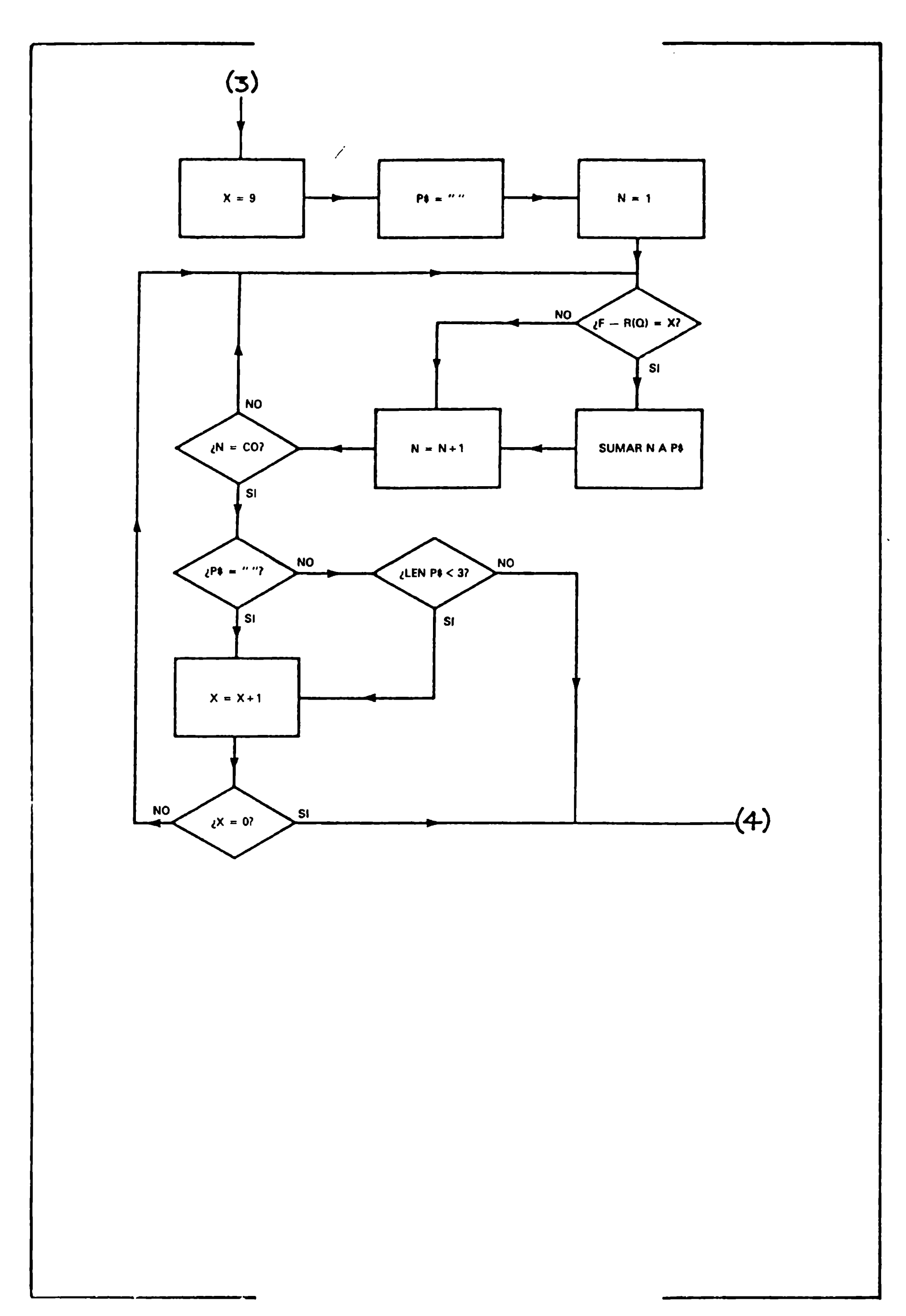
SU UNICA OPCION ES EL...



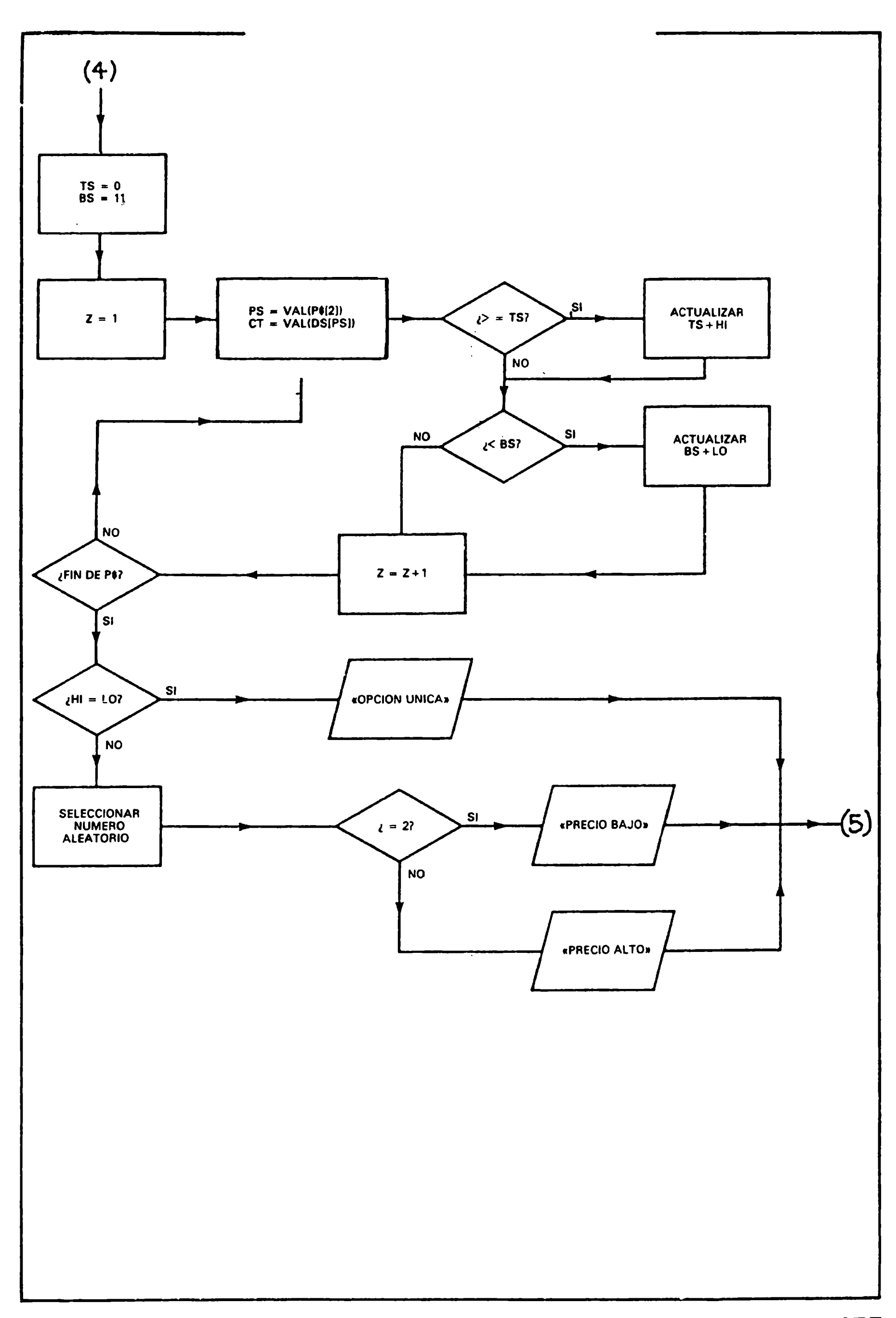
INICIALIZAR MATRICES
ELEGIR PALABRAS PARA LAS PREGUNTAS
CONJUGAR Y AÑADIR ESPACIO
IMPRIMIR PREGUNTA
INTRODUCIR RESPUESTA
AÑADIR ESPACIO Y REINICIALIZAR VARIABLES
¿HAY COMA?
SI
NO
¿HAY «Y» O «PERO»?
SI
NO
¿LA PRIMERA PALABRA ES «NO»?
SI
NO
SUSTRACCION DE LA REGLA
SUMA A LA REGLA
CORTE DELANTE DE COMA
¿HAY «SI»?
SI
NO
RU = RU+1
LD = 1
¿HAY «NO»?
SI
NO
NP = NP+1
LD = −1
¿HAY DOBLE NEGACION?
NO
SI
DECREMENTO DE RU
LD = −1
INCREMENTO DE RU
LD = 1
(1)



(1)
¿VERBOS FAVORABLES?
SI
LD = LD*1
NO
¿VERBOS DESFAVORA-BLES?
SI
LD = LD*–1
NO
¿OBJETO?
SI
RU = RU + LD
OF = N
OM = OM + 1
NO
¿ADVERBIO?
SI
RU = RU + LD
NO
¿ADVERBIO DESFAVORA-BLE?
SI
LD = LD*–1
RU = RU + LD
NO
¿ADJETIVO?
SI
RU = RU + LD
NO
¿ADJETIVO DESFAVORA-BLE?
SI
LD = LD*1
RU = RU + LD
NO
¿HM < 2?
SI
«BARATO»
NO
«CARO»
(2)
(2)
¿M < 1?
SI
«PERDON NO ENTIENDO»
NO
¿OM > 1?
SI
«CADA COSA A SU TIEMPO»
NO
¿SEÑALIZAR CARACTERISTICA?
ACTUALIZAR REGLA
NO
ACTUALIZAR REGLA OBJETO
ACTUALIZAR COSTE Y BENEFICIO TOTALES
«EXCUSAS»
SI
¿TP BAJO?
NO
ESTADO DEL CREDITO
SI
¿TC ALTO?
NO
(3)



(3)
X = 9
P$ = " "
N = 1
¿F – R(Q) = X?
NO
SI
SUMAR N A P$
N = N+1
¿N = CO?
NO
SI
¿P$ = " "?
NO
SI
¿LEN P$ < 3?
NO
SI
X = X+1
¿X = 0?
NO
SI
(4)


(4)
TS = 0
BS = 11
Z = 1
PS = VAL(P$(2))
CT = VAL(DS(PS))
¿> = TS?
SI
NO
ACTUALIZAR
TS + HI
¿< BS?
NO
SI
ACTUALIZAR
BS + LO
NO
¿FIN DE P$?
Z = Z+1
SI
¿HI = LO?
SI
«OPCION UNICA»
NO
SELECCIONAR
NUMERO
ALEATORIO
¿ = 2?
SI
«PRECIO BAJO»
(5)
NO
«PRECIO ALTO»

EL VENDEDOR

```
   1 REM RUTINAS PARA LOS SIGNOS
 INICIALES DE ADMIRACION E INTER
ROGACION (GRAFICOS DEL USUARIO)
   3 RESTORE 3: FOR I=1 TO 6: RE
AD A: POKE USR "A"+I,A: NEXT I:
DATA 4,0,4,4,4,4
   4 PRINT
   5 RESTORE 5: FOR N=1 TO 7: RE
AD B: POKE USR "I"+N,B: NEXT N:
DATA 2,0,2,14,16,17,14
   6 REM ***********************
   8 GO TO 9999
  10 BEEP .01,0: LET WL=LEN (J$)
: FOR I=FT TO LEN (I$)-WL+1: IF
I$(I TO (I+WL-1))=J$ THEN  LET S
P=I: RETURN
  20 NEXT I: LET SP=0: RETURN
 940 DATA "BASIC","GRAFICO","SON
IDO","TECLADO","FUNCION","MEMORI
A","CINTA MAGNETICA","MICRODRIVE
","DISCO","PROGRAMA","CARTUCHO",
"JOYSTICK","ENSAMBLADOR","CENTRO
NICS","RS232","EXPANSION","RED L
OCAL","16 BITS"
 950 DATA "BUEN","EXCELENTE","SU
PERIOR","MAGNIFIC","BUENA CLASE"
,"RAPID","EFICIENTE","ESENCIAL",
"MUCH","BIEN","NATURAL"
 955 DATA "REAL","MUY","FRECUENT
E","NECESARI","FIABLE"
 960 DATA "MAL","TRASTO","POBRE"
,"LENT","INEFICIENTE","POC","EL
MENOS","LA MENOS","PEOR"
 980 DATA "NUNCA","INFRECUENTE"
 990 DATA "QUIERO","DESE","NECES
IT","GUSTA","QUER"
1000 DATA "ODI","DETEST","ABORRE
","DESPRECI"
1010 DATA "&UN BUEN BASIC ",5,2,
```

```
"@UNOS BUENOS GRAFICOS ",7,2,"&E
L SONIDO ",6,2,"&UN BUEN TECLADO
 ",4,2,"@LAS TECLAS DE FUNCION "
,1,5,"&UNA GRAN MEMORIA ",3,6,"&
UN INTERFACE PARA CINTA ",2,2,"@
UNOS BUENOS MICRODRIVES ",2,4,"@
LOS DISCOS ",5,8,"@MUCHOS PROGRA
MAS ",0,9,"&UN PORT PARA CARTUCH
OS ",1,6,"&UN PORT PARA JOYSTICK
 ",1,7,"&UN ENSAMBLADOR ",2,1,"&
UN PORT CENTRONICS ",2,5,"&UN RS
232 ",2,6,"&UN EQUIPO EXPANDIBLE
 ",2,9,"&UN EQUIPO CAPAZ DE FORM
AR REDES ",3,4,"&UNA CPU DE 16 B
ITS ",1,7
1015 REM ************************
1020 REM LA LETRA "I" QUE ANTECE
DE CADA DATO EN LA LINEA 1030 CO
RRESPONDE AL SIGNO DE  INTERROGA
CION Y SE HA DE PULSAR EN MODO G
RAFICO
1025 REM ************************
1030 DATA "I LE GUSTARIA","I DES
EARIA","I QUE LE PARECE","I NECE
SITA USTED","I /*IMPORTANTE"
1032 DATA "BARAT","NO COSTOS"
1033 DATA "CAR","COSTOS"
1100 DATA "JNC PC","KNACT SERIOU
S","CLEARSIN MT","ACHRON ILLUSIO
N","BANANA IIE","SI ELITE","COLE
CTOVISION CABBAGE","CANDY COLOUR
ED COMPUTER","COMANDEAR 64","ATR
IA 600GT"
1110 DATA 6,6,9,8,3,8,5,7,2,1
1120 DATA 8,7,9,7,5,8,5,6,8,8
1130 DATA 8,6,9,6,2,8,5,4,9,8
1140 DATA 9,8,7,6,5,7,5,2,7,5
1150 DATA 8,8,7,0,0,7,2,0,7,0
1160 DATA 8,8,8,3,4,8,5,2,6,2
1170 DATA 8,8,8,7,6,8,5,7,5,5
1180 DATA 0,0,9,0,0,0,5,0,0,0
```

```
1190 DATA 9,8,9,5,3,7,5,4,6,7
1200 DATA 9,8,6,5,0,2,1,9,9,7
1210 DATA 7,0,7,0,3,7,7,8,6,7
1220 DATA 7,0,7,0,5,4,7,7,7,7
1230 DATA 0,0,0,6,0,0,0,0,0,0
1240 DATA 7,7,7,0,0,0,0,0,0,0
1250 DATA 6,6,6,0,6,6,6,6,2,6
1260 DATA 8,8,7,4,7,0,5,3,2,6
1270 DATA 8,8,9,1,0,0,0,0,0,0
1280 DATA 9,9,9,0,0,0,9,0,0,0
1290 DATA 9,9,9,0,0,0,0,0,0,0
1300 DATA 9,7,1,2,4,0,0,6,6,5
1350 DATA 10,9,7,3,8,4,6,5,2,1
1390 DATA "CREO QUE SE ESTA SALI
ENDO DE SUS POSIBILIDADES","ESTA
 ESPECIFICACION SE SALE DE SU LI
MITE DE CREDITO","NO CREO QUE PU
EDA PAGARSE TALES LUJOS"
1400 DATA "PERDONEME,PERO OIGO S
ONAR EL TELEFONO","TENGO UNA CIT
A URGENTE","CERRAMOS DENTRO DE C
INCO MINUTOS"
1440 DATA "SI PUEDE PAGARLO QUE
LE PARECE  EL ORDENADOR","UNA BU
ENA COMPRA ES EL ORDENADOR"
1450 DATA "SI DESEA UNA BUENA MA
QUINA        DEBE PROBAR EL ORDEN
ADOR","ACTUALMENTE NO SE PUEDE S
UPERAR EL ORDENADOR","SI DESEA U
N LUJO CONSIDERE EL ORDENADOR"
2000 LET PH=INT (RND*QP)+1: REST
ORE 1030: FOR N=1 TO PH: READ R$
: NEXT N
2100 LET I$=R$: LET J$="/": LET
FT=1: GO SUB 10: RESTORE 1010: F
OR N=1 TO Q: READ Q$,CR,PR: NEXT
 N
2200 IF SP<>0 THEN  IF Q$(1)="@"
 THEN  LET R$=R$( TO SP-1)+"SON"
+R$(SP TO )+"S"
3000 IF SP<>0 THEN  IF Q$(1)="&"
```

```
 THEN  LET R$=R$( TO SP-1)+"ES"+
R$(SP TO )
4000 LET I$=R$: LET J$="*": LET
FT=1: GO SUB 10: IF SP=0 THEN  G
O TO 4400
4200 LET R$=R$( TO SP-2)+" "+Q$(
2 TO )+R$(SP+1 TO ): GO TO 5000
4400 LET R$=R$+" "+Q$(2 TO )
5000 PRINT : PRINT : PRINT R$;"?
": INPUT N$: PRINT N$
5010 LET LO=0: LET TC=0: LET TP=
0: LET LD=1: LET OF=0: LET FS=1:
 LET NP=0: LET RU=0: LET M=0: LE
T OM=0: LET S1=0: LET S2=0
5020 LET I$=N$: LET J$=",": LET
FT=1: GO SUB 10: LET CM=SP
5030 IF CM=0 THEN  GO TO 5110
5040 LET I$=N$: LET J$="Y": LET
FT=1: GO SUB 10: LET S1=SP
5050 LET I$=N$: LET J$="PERO": L
ET FT=1: GO SUB 10: LET S2=SP
5060 IF S1+S2=0 THEN  GO TO 5100
5070 IF N$( TO 2)<>"NO" THEN  GO
 TO 5090
5080 LET R(Q)=R(Q)-1: LET I(Q)=I
(Q)-CR: LET J(Q)=J(Q)-PR: GO TO
5100
5090 LET R(Q)=R(Q)+1: LET I(Q)=I
(Q)+CR: LET J(Q)=J(Q)+PR
5100 LET N$=N$(CM TO )
5110 LET I$=N$: LET J$="SI": LET
 FT=FS: GO SUB 10
5120 IF SP>0 THEN  LET RU=RU+1:
LET LD=1: LET M=1: LET FS=SP+1:
GO TO 5110
5130 LET I$=N$: LET J$="NO": LET
 FT=FS: GO SUB 10
5140 IF SP>0 THEN  LET LD=-1: LE
T M=1: LET FS=SP+1: LET NP=NP+1:
 GO TO 5130
5180 RESTORE 1000: FOR N=0 TO DL
```

```
5190 LET I$=N$: READ J$: LET FT=
1: GO SUB 10
5200 IF SP>0 THEN  LET LD=LD*-1:
 LET M=1: LET NP=NP+1
5210 NEXT N
6000 RESTORE 990: FOR N=0 TO LI
6010 LET I$=N$: READ J$: LET FT=
1: GO SUB 10: IF SP=0 THEN  GO T
O 6030
6020 LET LD=LD*1: LET M=1
6030 NEXT N
6040 IF NP=0 THEN  GO TO 6100
6050 IF INT (NP/2)=NP/2 THEN  LE
T RU=RU+1: LET LD=1: GO TO 6100
6060 LET RU=RU-1: LET LD=-1
6100 RESTORE 940: FOR N=1 TO OB
6110 LET I$=N$: READ J$: LET FT=
1: GO SUB 10
6120 IF SP>0 THEN  LET RU=RU+LD:
 LET OF=N: LET M=1: LET OM=OM+1
6130 NEXT N
6200 RESTORE 950: FOR N=0 TO AV+
AJ
6210 LET I$=N$: READ J$: LET FT=
1: GO SUB 10: IF SP=0 THEN  GO T
O 6230
6220 LET RU=RU+LD: LET M=1
6230 NEXT N
6300 RESTORE 960: FOR N=0 TO NV+
NJ
6310 LET I$=N$: READ J$: LET FT=
1: GO SUB 10: IF SP=0 THEN  GO T
O 6330
6320 LET LD=LD*-1: LET RU=RU+LD:
 LET M=1
6330 NEXT N
6400 RESTORE 1032: FOR N=0 TO HM
6410 LET I$=N$: READ J$: LET FT=
1: GO SUB 10: IF SP=0 THEN  GO T
O 6440
6412 REM ***********************
```

```
6414 REM LA LETRA "A" QUE PRECE
DE A BARATO EN LAS LINEAS 6420 Y
6430 ES EL SIGNO DE ADMIRACION
DE ENTRADA Y SE HA DE PULSAR EN
MODO GRAFICO
6416 REM ************************
6420 LET XX=N: IF XX<2 THEN  PRI
NT '"AABARATO Y MALO ES TODO UNO
 !!": GO TO 6500
6430 PRINT '"AMAS BIEN CARO!": G
O TO 6450
6440 NEXT N
6500 IF M<1 THEN  PRINT "PERDON,
PERO NO LE ENTIENDO": GO TO 2000
6510 IF OM>1 THEN  PRINT "APOR F
AVOR,CADA COSA A SU TIEMPO!": GO
 TO 2000
6520 IF OF>0 THEN  LET R(OF)=R(O
F)+RU: LET I(OF)=I(OF)+(CR*RU):
LET J(OF)=J(OF)+(PR*RU): GO TO 6
550
6540 LET R(Q)=R(Q)+RU: LET I(Q)=
I(Q)+(CR*RU): LET J(Q)=J(Q)+(PR*
RU)
6545 PRINT
6550 FOR N=1 TO R: PRINT TAB (N-
1)*3;R(N);"  ";: LET TC=TC+I(N):
LET TP=TP+J(N): NEXT N: PRINT :
PRINT : FOR N=1 TO R: PRINT TAB
(N-1)*3;I(N);"  ";: NEXT N: PRINT
 : PRINT : FOR N=1 TO R: PRINT T
AB (N-1)*3;J(N);"  ";: NEXT N: PR
INT
6600 IF TC>BB THEN  LET PT=RND*C
S+0.5: RESTORE 1390: FOR N=0 TO
PT: READ Y$: NEXT N: PRINT Y$
6610 IF TP<Q THEN  LET TX=RND*EX
+0.5: RESTORE 1400: FOR N=0 TO T
X: READ X$: NEXT N: PRINT 'X$
6620 IF TC>BB THEN  LET PT=RND*C
S: RESTORE 1390: FOR N=0 TO PT:
```

```
READ Y$: NEXT N: PRINT Y$
7000 LET P$="": FOR X=9 TO 0 STE
P -1: RESTORE 1100+(Q*10): FOR N
=1 TO CO: READ F: IF F-R(Q)=X TH
EN  LET P$=P$+STR$ (N)
7010 NEXT N: IF LEN (P$)<=3 THEN
  NEXT X
7200 LET TS=0: LET BS=11: FOR Z=
1 TO LEN (P$): LET PS=VAL (P$(Z)
): LET CT=VAL (D$(PS)): IF CT>TS
 THEN  LET TS=CT: LET HI=PS
7220 IF CT<BS THEN  LET BS=CT: L
ET LO=PS
7230 NEXT Z: RESTORE 1100: IF HI
=LO THEN  PRINT "SU UNICA OPCION
 ES EL": FOR Z=1 TO HI: READ Z$:
 NEXT Z: PRINT Z$: PRINT ''"PULS
AR PARA SEGUIR": PAUSE 0: CLS :
GO TO 9000
7250 LET SE=INT ((RND*3)+1): LET
 SL=INT (RND*3)+1: IF SE=2 THEN
 GO TO 7290
7280 RESTORE 1450: FOR Z=1 TO SL
: READ Z$: NEXT Z: PRINT 'Z$,: R
ESTORE 1100: FOR Z=1 TO HI: READ
 Z$: NEXT Z: PRINT Z$: PRINT ''"
PULSE PARA SEGUIR": PAUSE 0: CLS
 : GO TO 9000
7290 RESTORE 1440: FOR Z=1 TO SL
: READ Z$: NEXT Z: PRINT 'Z$,: R
ESTORE 1100: FOR Z=1 TO LO: READ
 Z$: NEXT Z: PRINT Z$: PRINT ''"
PULSE PARA SEGUIR": PAUSE 0: CLS

9000 LET Q=Q+1: IF Q<FE+1 THEN
GO TO 2000
9010 STOP
9999 LET FE=18: LET R=FE: LET OB
=R-1: LET LI=3: LET DL=3: LET AJ
=8: LET AV=4: LET NJ=8: LET NV=1
: LET HM=3: LET QP=5: LET BB=50:
```

```
 LET CO=9: LET CT=9: LET CS=2: L
ET EX=2: LET HI=2: LET LO=2: LET
 Q=1: DIM R(R): DIM I(R): DIM J(
R): LET D$="9862735419": GO TO 2
000
```

Comentarios

Línea 9999: Contiene la rutina de iniciación de variables.
Líneas 10—20: Contiene la rutina de identificación (INSTR).
Línea 940: Contiene las palabras claves.
Línea 950: Contiene adjetivos.
Línea 955: Contiene adverbios.
Línea 960: Contiene adjetivos negativos (de rechazo).
Línea 980: Contiene adverbios negativos.
Línea 990: Contiene verbos favorables.
Línea 1000: Contiene verbos desfavorables.
Línea 1010: Contiene palabras para las preguntas (características) con los valores de coste y beneficio.
Línea 1030: Contiene frases interrogativas.
Línea 1032: Contiene palabras para expresiones de coste bajo.
Línea 1033: Contiene palabras para expresiones de coste alto.
Línea 1100: Contiene nombres de ordenadores (simulados).
Líneas 1110—1350: Contiene valores de las características de cada ordenador.
Línea 1390: Contiene mensajes sobre el crédito del comprador.
Línea 1400: Contiene excusas.
Línea 1440: Contiene mensajes de precios bajos.
Línea 1450: Contiene mensajes de precios elevados.
Líneas 2000—4400: Selecciona las palabras que se van a utilizar en la pregunta siguiente y prepara una correcta concordancia gramatical.
Líneas 5000—5010: Presenta la pregunta y recoge la respuesta entrada (INPUT); restablece variables.
Líneas 5020—5030: Busca una coma.

Líneas 5040—5060: Busca las conjunciones «Y» y «PERO», y si no las encuentra, el programa salta a la línea 5100.

Líneas 5070—5080: Actualiza negativamente la regla analizada si se encuentra «Y» o «PERO» y la primera palabra es «NO».

Línea 5090: Actualiza positivamente la regla analizada si se encuentra «Y» o «PERO» y la primera palabra no es «NO».

Línea 5100: Elimina todo lo que precede a la coma.

Líneas 5110—5170: Busca el «SI», el «NO» y el N'T actualiza adecuadamente la regla de decisión.

Líneas 5180—5210: Busca verbos desfavorables.

Líneas 6000—6030: Busca verbos favorables.

Línea 6040: Comprueba que no haya negaciones.

Línea 6050: Comprueba si hay doble negación.

Líneas 6100—6330: Busca los objetos, adjetivos y adverbios.

Líneas 6400—6440: Busca identidades con palabras claves de precios altos y bajos.

Línea 6500: Comprueba si no ha habido identidad y lo indica con el mensaje correspondiente.

Línea 6510: Comprueba si hay más de un objeto.

Líneas 6520—6540: Actualiza la regla analizada, u otra, si el objeto se adapta a la pregunta que se analiza.

Línea 6550: Presenta en pantalla las reglas y actualiza los valores de coste y de beneficio totales.

Línea 6600: Presenta en pantalla una advertencia si el gasto es demasiado alto.

Línea 6610: Presenta una excusa para determinar si el beneficio parece demasiado bajo.

Líneas 7000—7010: Busca los ordenadores que se adaptan a los requisitos del comprador.

Líneas 7200—7210: Escoge los ordenadores de mayor y menor precio que se adaptan a la especificación.

Línea 7220: Comprueba si sólo se ha seleccionado un ordenador.

Líneas 7250—7290: Presenta en pantalla el nombre del ordenador más caro o el del más barato.

Línea 9000: Actualiza la característica que se va a analizar y realiza el retorno para otra respuesta de entrada.

El lector ante sus creaciones en Inteligencia Artificial

La Inteligencia Artificial es un tema fascinante y confiamos haber proporcionado suficiente información para que el lector pueda iniciar sus propios experimentos en este área. Verdaderamente hemos disfrutado con nuestras propias exploraciones mientras preparábamos el libro pero hemos empezado a preguntarnos cuánto tiempo pasará antes que alguien se ingenie un programa de un sistema experto capaz de escribir obras completas.

OBRAS DE EDICIONES TECNICAS REDE

ALARMA ELECTRÓNICA (Libro n.º 102)
150 págs., 91 figs.

ALTA FIDELIDAD A BAJO COSTE (Libro n.º 87)
212 págs., 117 figs.

AUDIO REPARACIÓN (Libro n.º 126)
Autor: Felipe Mor. 170 págs., 187 figs.

AUTOMATISMOS DE FÁCIL CONSTRUCCIÓN (Libro n.º 107)
128 págs., 89 figs.

BIOELECTRÓNICA (Libro n.º 149)
132 págs., 72 figs.

CIRCUITOS COMPROBADOS-I:
AUDIO-1 (Libro n.º 111)
84 págs., 100 figs.

CIRCUITOS COMPROBADOS-II:
MONTAJES PRÁCTICOS (Libro n.º 115)
84 págs., 121 figs.

CIRCUITOS COMPROBADOS-III:
PRÁCTICA DIGITAL (Libro n.º 118)
80 págs., 134 figs.

CIRCUITOS COMPROBADOS-IV:
CIRCUITOS INTEGRADOS-1 (Libro n.º 128)
84 págs., 120 figs.

CIRCUITOS COMPROBADOS-V:
AUDIO-2 (Libro n.º 131)
88 págs., 100 figs.

CIRCUITOS COMPROBADOS-VI:
JUEGOS ELECTRÓNICOS-1 (Libro n.º 132)
80 págs., 100 figs.

CIRCUITOS COMPROBADOS-VII:
ANTI-ROBO (Libro n.º 135)
76 págs., 114 figs.

CIRCUITOS COMPROBADOS-VIII:
JUEGOS ELECTRÓNICOS-2 (Libro n.º 141)
80 págs., 96 figs.

CIRCUITOS COMPROBADOS-IX:
AUDIO-3 (Libro n.º 143)
82 págs., 70 figs.

CIRCUITOS COMPROBADOS-X:
TELEMANDO (Libro n.º 146)
80 págs., 118 figs.

CIRCUITOS COMPROBADOS-XI:
COMPROBADORES (Libro n.º 152)
84 págs., 111 figs.

CIRCUITOS COMPROBADOS-XII:
AUDIO-4 (Libro n.º 154)
80 págs., 118 figs.

CIRCUITOS COMPROBADOS-XIII:
ELECTRÓNICA EN EL AUTOMÓVIL (Libro n.º 158)
84 págs., 124 figs.

CIRCUITOS COMPROBADOS-XIV:
LUCES SICODÉLICAS Y JUEGOS LUMINOSOS
(Libro n.º 160)
80 págs., 100 figs.

CIRCUITOS COMPROBADOS-XV:
CIRCUITOS INTEGRADOS-2 (Libro n.º 164)
80 págs., 102 figs.

CIRCUITOS COMPROBADOS-XVI:
ELECTRONICA PARA TRENES DE JUGUETE
(Libro n.º 181)
82 págs., 102 figs.

CIRCUITOS ELECTRÓNICOS CONTROLADOS POR ORDENADOR (Libro n.º 194)
136 págs., 29 figs.

COMODIDADES ELECTRÓNICAS (Libro n.º 99)
124 págs., 75 figs.

COMUNICACIÓN INSTANTÁNEA (Libro n.º 109)
114 págs., 54 figs.

CON UN TRANSISTOR, MÚLTIPLES MONTAJES COMPROBADOS (Libro n.º 120)
118 págs., 62 figs.

CON DOS TRANSISTORES, MÚLTIPLES MONTAJES COMPROBADOS (Libro n.º 124)
140 págs., 67 figs.

CON TRES TRANSISTORES, MÚLTIPLES MONTAJES COMPROBADOS (Libro n.º 125)
132 págs., 64 figs.

CONSTRUCCIÓN DE CAJAS ACÚSTICAS (Libro n.º 138)
122 págs., 79 figs.

CONTRAESPIONAJE ELECTRÓNICO (Libro n.º 93)
115 págs., 50 figs.

CURSO RÁPIDO DE TECNOLOGÍA DIGITAL (Libro n.º 202)
Autor: Louis E. Frenzel jr. 228 págs., tamaño 28 × 22 cms.

ELECTRÓNICA AL SERVICIO DEL AUTOMOVILISTA
(Libro n.º 122)
146 págs., 68 figs.

ELECTRÓNICA EN LA FOTOGRAFÍA (Libro n.º 121)
126 págs., 52 figs.

ESPIONAJE ELECTRÓNICO (Libro n.º 84)
152 págs., 71 figs.

ESQUEMARIOS DE TV B/N
Tamaño: 27 × 21 cms.

ESQUEMARIO TV/I (libro n.º 21)
ESQUEMARIO TV/II (libro n.º 22)
ESQUEMARIO TV/III (libro n.º 55)
ESQUEMARIO TV/IV (libro n.º 67)
ESQUEMARIO TV/V (libro n.º 76)
ESQUEMARIO TV/VI (libro n.º 81)
ESQUEMARIO TV/VII (libro n.º 88)
ESQUEMARIO TV/VIII (libro n.º 94)
ESQUEMARIO TV/IX (libro n.º 100)
ESQUEMARIO TV/X (libro n.º 105)
ESQUEMARIO TV/XI (libro n.º 113)
ESQUEMARIO TV/XII (libro n.º 127)
ESQUEMARIO TV/XIII (libro n.º 140)
ESQUEMARIO TV/XIV (libro n.º 155)

ESQUEMARIO DE MAGNETÓFONOS Y CASSETTES-I
(Libro n.º 85)

ESQUEMARIO DE MAGNETÓFONOS Y CASSETTES-II
(Libro n.º 103)

ESQUEMARIO DE MAGNETÓFONOS Y CASSETTES-III
(Libro n.º 123)

ESQUEMARIO DE MAGNETÓFONOS Y CASSETTES-IV
(Libro n.º 130)

ESQUEMARIO DE MAGNETÓFONOS Y CASSETTES-V
(Libro n.º 133)

ESQUEMARIO DE MAGNETÓFONOS Y CASSETTES-VI
(Libro n.º 139)

ESQUEMARIO DE MAGNETÓFONOS Y CASSETTES-VII
(Libro n.º 144)

ESQUEMARIO DE MAGNETÓFONOS Y CASSETTES-VIII
(Libro n.º 159)

ESQUEMARIO DE TVC-I (Libro n.º 112)

ESQUEMARIO DE TVC-II (Libro n.º 114)

ESQUEMARIO DE TVC-III (Libro n.º 116)

ESQUEMARIO DE TVC-IV (Libro n.º 117)

ESQUEMARIO DE TVC-V (Libro n.º 119)

ESQUEMARIO DE TVC-VI (Libro n.º 129)

ESQUEMARIO DE TVC-VII (Libro n.º 134)

ESQUEMARIO DE TVC-VIII (Libro n.º 136)

ESQUEMARIO DE TVC-IX (Libro n.º 137)

ESQUEMARIO DE TVC-X (Libro n.º 142)

ESQUEMARIO DE TVC-XI (Libro n.º 145)

ESQUEMARIO DE TVC-XII (Libro n.º 150)

ESQUEMARIO DE TVC-XIII (Libro n.º 157)

ESQUEMARIO DE TVC-XIV (Libro n.º 161)

ESQUEMARIO DE TVC-XV (Libro n.º 162)

ESQUEMARIO DE TVC-XVI (Libro n.º 165)

ESQUEMARIO DE TVC-XVII (Libro n.º 167)

ESQUEMARIO DE TVC-XVIII (Libro n.º 168)

ESQUEMARIO DE TVC-XIX (Libro n.º 170)

ESQUEMARIO DE TVC-XX (Libro n.º 171)

ESQUEMARIO DE TVC-XXI (Libro n.º 176)

ESQUEMARIO DE TVC-XXII (Libro n.º 180)

ESQUEMARIO DE TVC-XXIII (Libro n.º 186)

ESQUEMARIO DE TVC-XXIV (Libro n.º 187)

ESQUEMARIO DE TVC-XXV (Libro n.º 192)

ESQUEMARIO DE TVC-XXVI (Libro n.º 199)

ESQUEMARIO DE VIDEO-1 (Libro n.º 174)

GRABADORES DOMÉSTICOS DE VIDEO.
GUÍA DE SERVICIO (Libro n.º 188)
Autor: Steve Beeching. 144 págs., 116 figs.
Tamaño: 27 × 21 cms.

GRABADORES DE VIDEO. GUÍA PRÁCTICA PARA PRINCIPIANTES (Libro n.º 204)
Autor: Eugene Trundle. 235 págs., 130 figs.

IMPROVISACIONES QUE DAN DINERO Y AHORRAN TIEMPO

Volumen I (libro n.º 48): 140 págs., 91 figs.
Volumen II (libro n.º 63): 126 págs., 77 figs.
Volumen III (libro n.º 80): 130 págs., 82 figs.
Volumen IV (libro n.º 98): 146 págs., 90 figs.
Volumen V (libro n.º 104): 138 págs., 91 figs.

INICIACIÓN A LA PRÁCTICA ELECTRÓNICA (I): FUNDAMENTOS (Libro n.º 201)
Autor: Felipe Mor. 236 págs., 98 figs.

INICIACIÓN A LA PRÁCTICA ELECTRÓNICA (II): COMPONENTES
Autor: Felipe Mor.

INICIACIÓN AL DISEÑO DE CIRCUITOS DE AUDIO (Libro n.º 148)
108 págs., 42 figs.

JUGUETES ELECTRÓNICOS (I) (Libro n.º 96)
122 págs., 61 figs.

JUGUETES ELECTRÓNICOS (II) (Libro n.º 106)
128 págs., 91 figs.

MAGNETÓFONOS Y CASSETTES (Libro n.º 151)
150 págs., 70 figs.

MANUAL DE CIRCUITOS INTEGRADOS TTL (Libro n.º 198)
Autor: Don Lancaster. 382 págs., 169 figs.

MÚSICA ELECTRÓNICA (Libro n.º 83)
150 págs., 70 figs.

ÓRGANOS ELECTRÓNICOS (Libro n.º 169)
Autor: A. Quilez. 206 págs., 118 figs.

PRÁCTICA DE LA CONSTRUCCIÓN E INSTALACIÓN DE ANTENAS (Libro n.º 92)
272 págs., 222 figs.

RECUPERACIÓN DE COMPONENTES ELECTRÓNICOS (Libro n.º 156)
166 págs., 91 figs.

REPARACIÓN DE ELECTRODOMÉSTICOS (Libro n.º 40)
265 págs., 167 figs.

REPARACIÓN TV-I (Libro n.º 77)
Autor: Felipe Mor. 300 págs., 472 figs.

REPARACIÓN TV-II (Libro n.º 110)
Autor: Felipe Mor. 304 págs., 470 figs.

REPARACIÓN TV COLOR (Libro n.º 153)
140 págs., 125 figs.

ROBÓTICA PRÁCTICA (Libro n.º 200)
Autor: José M.ª Angulo. 376 págs., 263 figs.

SEGURIDAD ELECTRÓNICA (Libro n.º 108)
120 págs., 60 figs.

SOLDADURA ELÉCTRICA (Libro n.º 147)
Autor: Felipe Mor. 104 págs., 69 figs.

TIRISTOR. APLICACIONES, CARACTERÍSTICAS, FUNCIONAMIENTO (Libro n.º 73)
Autor: R. Swoboda. 104 págs., 50 figs.

TV COLOR BÁSICA (Libro n.º 166)
170 págs., 84 figs., 12 láminas color.

60.000 TRANSISTORES (Libro n.º 163)
404 páginas.

MICROINFORMÁTICA

APLICACIONES DEL CÓDIGO MÁQUINA PARA EL ZX-SPECTRUM (Libro n.º 190)
Autor: David Laine. 170 págs.

PROGRAMACIÓN EN CÓDIGO MÁQUINA PARA EL ZX-81 Y EL SPECTRUM (Libro n.º 175)
Autor: Joan Sales Roig. 158 págs.

CÓDIGO MÁQUINA SIMPLIFICADO PARA EL ZX-SPECTRUM (Programación avanzada) (Vol. II)
Autor: Paul Holmes.

COMMODORE-64 «Una selección de juegos» (Libro n.º 183)
Autor: William A. Roberts. 98 págs.

LA MEJOR PROGRAMACIÓN DEL DRAGÓN POR LA PRÁCTICA (Libro n.º 184)
Autores: Keith y Steven Brain. 252 págs., 50 figs.

GUÍA PRÁCTICA DE BASIC DEL ZX-81 Y DEL SPECTRUM (Libro n.º 172)
Autor: R. Rovira Soligó. 148 págs., 31 figs.

EL CÓDIGO MÁQUINA DEL SPECTRUM SIMPLIFICADO (Volumen I) (Libro n.º 185)
Autor: James Walsh. 235 págs., 28 figs.

ACCESO RÁPIDO AL VIC-20 (Libro n.º 173)
Autor: Tim Hartnell, 158 págs.

TÉCNICA Y PRÁCTICA DE LOS JUEGOS DE AVENTURAS DEL ZX-SPECTRUM (Libro n.º 189)
Autores: Tony Bridge y Roy Carnell. 186 págs.

ZX-MICRODRIVE (Libro n.º 179)
Autor: Andrew Pennell. 177 págs.

LA MEJOR PROGRAMACIÓN DEL ZX-SPECTRUM POR LA PRÁCTICA (Libro n.º 177)
Autores: Tim Hartnell y Dilwyn Jones. 266 pág.

60 PROGRAMAS COMPLETOS PARA SPECTRUM (Libro n.º 178)
Autor: David Harwood. 133 págs.

MSX-SELECCIÓN DE PROGRAMAS (Libro n.º 191)
Autor: Vince Apps. 166 págs.

GUÍA PRÁCTICA PARA LA PROGRAMACIÓN CREATIVA DEL SPECTRUM (Libro n.º 195)
Autor: Mike James. 302 págs.

LA MEJOR PROGRAMACIÓN DEL COMMODORE 64 POR LA PRÁCTICA (Libro n.º 193)
Autores: Peter Lupton y Frazer Robinson.
302 págs., 70 figs.

GUÍA PRÁCTICA DEL PROGRAMADOR EN CÓDIGO MÁQUINA DEL SPECTRUM (Libro n.º 197)
Autor: R. Ross-Langley. Tamaño: 27 × 21 cms. 216 págs.

INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL CON EL SPECTRUM
Autores: Keith y Steven Brain. 188 págs.

VARIOS

TIROS DE COMBATE Y DEFENSA PERSONAL
(Libro n.º 182)
Autor: Siegfried F. Hübnner. 260 págs., 308 figs.

ARMAS DE COMBATE (Libro n.º 196)
Autor: Dominique Venner. 330 págs.


ediciones
técnicas
REDE
BARCELONA
(ESPAÑA)